Michal Varga, Josef Rak, Dusan Savic, Zlatko Stapic

Grundlagen der objektorientierten Programmierung

-

Ein praktischer Leitfaden für Lehrer

Eine Übersetzung aus dem Originaldokument: GUIDE FOR CURRENT TEACHERS OF THE HIGH SCHOOLS

Projekt	Object Oriented Programming for Fun
Projektakronym	OOP4FUN
Vertragsnummer	2021-1-SK01-KA220-SCH-00027903
Projektkoordinator	Universität Žilina (Slowakei)
Projektpartner	Universität Zagreb (Kroatien) Gymnasium Ivanec (Kroatien) Universität Pardubice (Tschechische Republik) Gymnasium, Pardubice, Dašická 1083 (Tschechische Republik) Business Academy Považská Bystrica (Slowakei) Hochschule für Technik und Wirtschaft Dresden (Deutschland) Gymnasium Dresden-Plauen (Deutschland) Universität Belgrad (Serbien) Gymnasium Ivanjica (Serbien)
Erscheinungsjahr	2024

Inhaltsverzeichnis

Tabellenverzeichnis

1. Einordnung des Buchs

Das Ziel dieses Buches ist es, einen Kurs mit Materialien vorzustellen, die Lehrern bei der Vorbereitung ihres Unterrichts helfen. Im Unterricht werden Schüler Programmieraufgaben nach dem Konzept der objektorientierten Programmierung (OOP) gemäß dem Light-OOP-Paradigma lösen.

Die Schülerinnen und Schüler lernen, vorgegebene Aufgaben auf kooperierende Objekte aufzuteilen; die Zuständigkeiten zu bestimmen; und das entworfene Modell zu implementieren. Für den Kurs sind keine vorherigen Programmierkenntnisse erforderlich. Es wird in der Programmiersprache Java unterrichtet. Wir verwenden die Greenfoot-Umgebung, die auf der Programmiersprache Java aufbaut. Java ist derzeit eine sehr beliebte und in der Praxis weit verbreitete Programmiersprache. Darüber hinaus strellt Greenfoot den framebasierten Quellcode-Editor in der Stride-Sprache bereit. Dies eröffnet Möglichkeiten für Lehrer, die in diesem Lehrplan vorgestellte Techniken mit Schülern jüngeren Alters anwenden möchten. Greenfoot ist visuell gestaltet und ermöglicht es von Anfang an, visualisierte Objekte zu erzeugen, die "lebendig" sind und mit denen interagiert werden kann. Dadurch kann die theoretische Einführung kurzgehalten werden und die Schüler können von Anfang an praktische Aufgaben lösen.

Der Kurs erklärt Light-OOP-Konzepte (wie z.B. Kapselung, Vererbung oder Assoziation) bei der Erstellung von Computerspielen, wobei diese Konzepte einfach und intuitiv angewendet werden. Der Prozess der Erstellung eines Computerspiels basiert auf Teamarbeit und nutzt praktisch Kenntnisse und Fähigkeiten aus anderen Bereichen der Informatik und zu verwandten Themen (Arbeit mit Multimedia und Bürosoftware). Die Gestaltung eines jeden Computerspiels ist offen genug, damit die Schülerinnen und Schüler das Spiel individuell und kreativ erweitern können. Darüber hinaus führt das Design zur richtigen Nutzung des erworbenen Wissens.

Das Buch konzentriert sich auf die Einführung eines innovativen Ansatzes zum Unterrichten des Programmierens, der auf der Lösung von Aufgaben mit dem Paradigma der objektorientierten Programmierung (OOP) basiert. OOP ist heutzutage das dominierende Paradigma für die Anwendungsentwicklung. Daher ist es für die Schüler angemessen, über Kenntnisse und Fähigkeiten in diesem Bereich zu verfügen. Im Buch wird eine Entwicklungsumgebung benutzt, die verschiedene Formen der Quellcodebearbeitung verwendet (rahmenbasiertes Bearbeiten in vereinfachter Form, sowie echtes Schreiben von Quellcode), was es ermöglicht, die Schüler auf verschiedenen Ebenen des technischen Vorwissens und der Aktivität zu unterrichten. Mit seiner Einfachheit und Klarheit unterstützt dieses Tool die schnelle und intuitive Auffassung der gelehrten Themen, was einen positiven Einfluss auf die Schüler und ihre Motivation hat.

Durch das Programmieren von interaktiven Spielen in einer grafischen Umgebung erwerben die Schüler Kenntnisse und Fähigkeiten. Sie werden dann in der Lage sein:

1. ein Problem zu identifizieren,
2. geeignete Objekte zur Lösung des identifizierten Problems (Objektzerlegung) zu identifizieren,
3. Klassen für die Objekte, sowie deren Attribute und Methoden zu programmieren,

4. Objektbeziehungen (Assoziation, Vererbung) zu erkennen und richtig zu nutzen,
5. einen Algorithmus zur Lösung von Problemen zu entwerfen und ihn auf kooperierende Objekte zu verteilen,
6. Quellcode-Elemente (Verzweigungen, Schleifen) zu verwenden, um den entworfenen Algorithmus zu implementieren,
7. Mittel zum Debuggen von Quellcode effektiv zu nutzen,
8. eine einfache Anwendung mit grafischer Oberfläche in der Greenfoot-Umgebung zu erstellen.

Die Lernergebnisse werden wie folgt zusammengefasst:

1. Verständnis der Grundprinzipien der objektorientierten Programmierung,
2. Verständnis der Grundlagen des Algorithmendesigns,
3. Verständnis der Syntax der Programmiersprache Java,
4. Fähigkeit der Analyse einer Programmausführung auf der Grundlage des Quellcodes,
5. die Möglichkeit, eigene Programme mit Hilfe von objektorientierter Programmierung zu erstellen.

Ein moderner Ansatz bei der Gestaltung von Vorlesungen, insbesondere für die Grund- und Oberstufe, besteht darin, Unterrichtsszenarien zu definieren und zu teilen. Unterrichtsszenarien (Teaching Scenarios, TS) werden als zeitgemäßer pädagogischer Ansatz verstanden, der eine Individualisierung des Unterrichtsprozesses ermöglicht, indem die unterschiedlichen Bedürfnisse der Studierenden berücksichtigt werden. Die TS-basierte Lehre konzentriert sich auf relevante Kenntnisse und Fähigkeiten für die Schüler, einschließlich derjenigen, die für die digitale Gesellschaft benötigt werden. Eine sorgfältige Planung von TS kann mögliche Fallstricke und Mängel beheben, die den Lehrprozess beeinflussen könnten.

Im Kontext von Bildung und Unterrichtsdesign stellen TS detaillierte Beschreibungen oder Erzählungen dar, die eine bestimmte Unterrichtssituation oder einen bestimmten Kontext umreißen. Diese Szenarien werden in der Lehrerausbildung häufig verwendet, um reale Unterrichtssituationen zu simulieren, und daher empfinden wir sie als das beste Werkzeug, um unsere innovativen Lehr- und Lernideen darzustellen. Da Unterrichtsszenarien in der Regel Informationen über die Lernziele, die zu vermittelnden Inhalte, die Eigenschaften der Lernenden, die angewandten Unterrichtsmethoden und die verwendeten Bewertungsstrategien enthalten, können sie auch mit den Elementen abgeglichen werden, die für unsere Lerndesign-Artefakte benötigt werden.

Um einen strukturierten Ansatz bei der Definition mehrerer Unterrichtsszenarien vorzubereiten, haben wir eine Vorlage definiert, die mit konkreten Daten zu einem bestimmten Unterrichtsszenario gefüllt wird. Die Vorlage enthält eine kurze Beschreibung, wie die einzelnen Elemente des TS definiert werden.

Tabelle 1. Vorlage zur Dokumentation von Unterrichtsszenarien

Titel	Geben Sie dem Lernszenario (Teaching Scenario, TS) einen anschaulichen und einprägsamen Titel.
Lernziele	Geben Sie die beabsichtigten Lernergebnisse klar an. Was sollten die Schüler am Ende des Szenarios wissen, verstehen oder in der Lage sein?
Zielgruppe	Geben Sie die Zielgruppe, die Klassenstufe und das geforderte Vorwissen an, für das das Lernszenario konzipiert ist.
Dauer des Szenarios	Schätzen Sie die Zeit, die zum Abschließen des Lernszenarios erforderlich ist, einschließlich spezifischer Zeitrahmen für verschiedene Aktivitäten. Z.B. Lehrereinführung (5 min), Recherche durch Schüler individuell (10 min), Programmierlösung im Team (20 min), Präsentation/Diskussion (10 min).
Materialien und Ressourcen	Listen Sie die Materialien, Ressourcen und Werkzeuge auf, die sowohl für Lehrer als auch für Schüler benötigt werden. Das kann Lehrbücher, Online- und Multimedia-Ressourcen, Software usw. umfassen.
Beschreibung	1. Stellen Sie das Lernszenario vor, erläutern Sie dessen Zweck und Relevanz. 2. Skizzieren Sie die Kernaktivitäten, an denen sich die Schüler beteiligen, um die Lernziele zu erreichen. Fügen Sie Details wie Diskussionen, praktische Aktivitäten, Gruppenarbeiten, Wettbewerbe usw. hinzu. 3. Legen Sie fest, wie die Studierenden organisiert werden, d.h. ob sie einzeln oder im Team arbeiten sollen. Wie groß werden die Teams sein? 4. Erklären Sie, an welchen Projekten/Problemen/Aufgaben die Schülerinnen und Schüler arbeiten. Es wird empfohlen, problembasierte oder projektbasierte Ansätze zu verwenden. Außerdem sollten reale Situationen widergespiegelt werden.

	5. Erklären Sie, wie Projekte/Probleme/Aufgaben den Studierenden (Teams) zugewiesen werden. 6. Wenn Sie in Teams arbeiten, geben Sie Details dazu an, wie sie zusammenarbeiten werden. 7. Geben Sie weitere Details zu den Aktivitäten an, an denen die Schüler teilnehmen müssen. Wenn z.B. Flipped Classrooms verwendet werden, geben Sie an, welcher Teil des gegebenen Themas die Schülerinnen und Schüler selbst recherchieren müssen.
Bewertung	Geben Sie Einzelheiten dazu an, wie die Leistungen und Kenntnisse der Schüler bewertet werden. 1. Wer wird die Schüler bewerten: (1) Lehrer, (2) Schüler ihre eigene Arbeit (Selbstbewertung), (3) Schüler untereinander (Peer-Bewertung) 2. Welche Bewertungskriterien werden herangezogen? Wie oft wird eine Bewertung durchgeführt, usw.?
Bereitstellung der Ergebnisse	Erklären Sie, wie die Schüler ihre Ergebnisse an Lehrer und Mitschüler weitergeben sollen. Beispielsweise können die Schüler (alle oder ein Teil) ihre Ergebnisse/Lösungen vor dem Lehrer und den Mitschülern präsentieren, um und danach Vergleiche und die Diskussion vorzunehmen.

Ein Tool zur Erstellung dementsprechender Unterrichtsszenarien wird unter folgender Webadresse angeboten: learning-design.eu

Unterstützende Materialien finden Sie auf der Moodle-Plattform: https://oop4fun.fon.bg.ac.rs/

2. Einführung in die Greenfoot-Programmierumgebung

Greenfoot ist eine visuelle 2D-Lernsoftware mit einem Code-Editor zum Erstellen von Spielen und Simulationen in der Programmiersprache Java. Greenfoot ist visuell und interaktiv. Die Programme werden in Standard-Java-Code programmiert. Dadurch entsteht eine Kombination aus Programmerstellung in einer traditionellen textbasierten Sprache und visueller Ausführung.

2.1. Spielerische Einarbeitung in die Spieleentwicklung

2.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 2. Spielerische Einarbeitung in die Spieleentwicklung

Titel	Spielerische Einarbeitung in die Spieleentwicklung
Lernziele	In dieser Sitzung haben die Studenten nicht nur Greenfoot erfolgreich installiert und die Fähigkeiten anhand von Beispielprojekten kennengelernt, sondern haben sich auch an kollaborativem, praktischem Spielen in der Entwicklungsumgebung beteiligt. Diese spielerische Einführung gibt den Anstoß für eine weitere Erkundung der Spieleentwicklung an und fördert Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren mit Greenfoot.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen.
Dauer des Szenarios	1. Einführung (5 Minuten) 2. Rush-Hour-Challenge (10 Minuten) 3. Spielen mit dem Lehrer (30 Minuten) 4. Teambildung und Projektzuweisung (5 Minuten) 5. Zusammenarbeit und Programmierung im Team (30 Minuten) 6. Peer-Review und Feedback (10 Minuten) 7. Hausaufgaben (30 Minuten) 8. Bewertung und Einstufung (30 Minuten)
Materialien und Ressourcen	Greenfoot-Webseite und Download-Anweisungen. Beispiele, die vom Lehrer vorbereitet wurden. Internetressourcen zur Identifizierung anderer Beispiele.
Beschreibung	In diesem 90-minütigen Unterrichtsszenario tauchen Schülerinnen und Schüler der Sekundarstufe mit Hilfe von Gamification, Spaß, Recherche und Teamwork in die Greenfoot-Welt ein. Nachdem die Lehrkraft die heutige Sitzung vorgestellt, die vorherige Sitzung reflektiert und herausfordernde Ziele gesetzt hat, beginnt die **Rush-Hour-Challenge**. Die Schüler erhalten die spielerische Aufgabe, Anweisungen zu finden,

	Greenfoot (ein für sie noch unbekanntes Entwicklungstool) herunterzuladen und auf ihren Computern zu installieren. Die ersten drei Schüler erhalten eine Belohnung (Punkte, Süßigkeiten o.ä.). Die zweite Überraschung für sie ist, dass sie in den nächsten 30 Minuten **mit dem Lehrer Spiele spielen** werden. Dies ist eine von Lehrern geleitete Sitzung zum Öffnen, Kompilieren und Ausführen von ein bis zwei einfachen Beispielprojekten (auf der einführenden bis mittleren Komplexitätsstufe). Dabei werden den Schülern die grundlegenden Elemente der Greenfoot-Entwicklungsumgebung und grundlegende Vorgehensweisen für den Umgang mit den Projektdateien und Programmelementen gezeigt. Anschließend werden die Schülerinnen und Schüler in **Teams** (je 3-4 Schülerinnen und Schüler) eingeteilt und erhalten eine einfache Aufgabe. Die Teams sollten in dem gegebenen Beispielprojekt etwas ändern, um das Spiel überraschend oder unterhaltsam zu gestalten. In dem Abschnitt **Teamzusammenarbeit und Codierung** (30 Minuten) können die Teams gemeinsam daran arbeiten, etwas in den gegebenen Beispielen zu ändern. Wenn sie den Code so weit kaputt machen, dass sie ihn selbst reparieren können, können sie den Lehrer um Hilfe bitten oder die "Startversion" erneut herunterladen. Dies wird ein gutes Beispiel dafür sein, warum wir beim Programmieren ein Versionskontrollsystem (GIT) verwenden sollten. Ein bis zwei Teams stellen ihre Arbeit zur Begutachtung und zum Feedback vor und besprechen die Ergebnisse mit der Lehrkraft. Als **Hausaufgabe** sollte jeder Schüler nach Beispielen für Greenfoot-Spiele suchen und eine Vorstellung seines Lieblingsspiels vorbereiten, indem er einen Link, eine Beschreibung, was sein Lieblingsbeispiel macht, und zwei bis drei Screenshots der Entwicklungsumgebung und des laufenden Spiels hochlädt. Im Rahmen der **Gamification** und Motivation durch **Wettbewerb** soll jeder Schüler für die drei besten Spiele abstimmen (es ist nicht erlaubt, für sein eigenes Spiel abzustimmen). Die Gewinner werden bekannt gegeben und belohnt (Abzeichen, Punkte, Punkte, Süßigkeiten usw.).
Bewertung	Diese Aktivität ermöglicht es den Lehrern, auf der Grundlage der Diskussionen und der Überwachung des Flipped Classroom und der Teamarbeit der Schüler Feedback zur Bewertung zu geben. Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Das Peer-Review-Assessment wird online im Rahmen einer Hausaufgabe durchgeführt. Dies erinnert die Schüler an wichtige Aspekte des Unterrichts und veranlasst, dass sie Greenfoot installieren. Durch die verschiedenen Beispiele erinnern sich die Schüler daran, was während des Unterrichts getan wurde, was die Gesamtleistung der Lernergebnisse erhöht.
Bereitstellung	Zur Weitergabe der Ergebnisse an Lehrende und Mitschüler wird ein übliches

der Ergebnisse	Lernmanagementsystem (z.B. Moodle) verwendet. Die Schülerinnen und Schüler können die Diskussion zum Thema in dem ihnen zur Verfügung gestellten Forum über das Lernmanagement-Tool fortsetzen.

2.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Einführung

Ziel: Einführung der Schüler in Programmiersprachen (visuell und textuell), integrierte Entwicklungsumgebungen und Quellcode.

Zu diskutierende Konzepte: Programmiersprachen, integrierte Entwicklungsumgebungen und Quellcode

Aktivität: Die Lehrkraft gibt eine kurze Einführung in die heutige Unterrichtsstunde. Die Lehrkraft führt Konzepte ein wie:

1. Programmiersprache
2. integrierte Entwicklungsumgebungen (Programmierwerkzeuge) und
3. Quellcode.

Der Lehrer stellt Beispiele für Quellcode in verschiedenen Programmiersprachen wie Java, C und Python vor, demonstriert aber auch Programme in Scratch oder einer anderen visuellen Sprache.

Der Lehrer geht nicht "in die Tiefe", sondern versucht, diese Konzepte anhand eines einfachen Beispiels zu erklären, indem er beispielsweise versucht, das Erlernen einer Programmiersprache mit dem Erlernen einer Mutter- oder Fremdsprache zu vergleichen.

1. Natürliche Sprachen haben ihre eigene Grammatik und Rechtschreibung. Das Gleiche gilt für Programmiersprachen. Beim Schreiben befolgen wir einige Grammatik- und Rechtschreibregeln, ähnlich wie wenn wir ein Programm in einer Programmiersprache schreiben.
2. Wenn wir eine Geschichte in unserer Muttersprache schreiben, verwenden wir ein Notizbuch, einen Stift als Werkzeug, das uns dabei hilft, ähnlich wie wenn wir ein Programm in einer Programmiersprache schreiben. Dann ist unser "Werkzeug" dafür eine integrierte Entwicklungsumgebung.
3. Das Ergebnis des Schreibens kann eine auf Papier geschriebene Geschichte sein, während das Ergebnis des Programmierens ein Programm ist, das wir erstellen und das wir als Quellcode bezeichnen.

Dem Lehrer steht es frei, eine Einleitung mit seinem eigenen Beispiel zu machen.

2. Rush-hour Challenge

Gegenstand: Vertraut machen mit Greenfoot und den Installationsanweisungen.

Zu besprechende Konzepte: Greenfoot, Installationsanleitung

Aktivität: Die Lehrkraft gibt den Schülerinnen und Schülern die Aufgabe, herauszufinden, was Greenfoot ist. Der Lehrer verlangt von den Schülern, dass sie die Installationsanleitung für Greenfoot finden. Die Schüler bearbeiten die Aufgabe, danach stellt der Lehrer den Schülern vor, wie sie die Installationsdatei für Greenfoot finden, herunterladen und ausführen können.

Anweisungen zum Herunterladen und Installieren von Greenfoot finden Sie auf der Moodle-Plattform.

3. Spielen mit dem Lehrer

Gegenstand: Starten verschiedener Greenfoot-Projekte.

Zu besprechende Konzepte: Greenfoot-Projekt (webbasierte und eigenständige Projekte)

Aktivität: Der Lehrer gibt den Schülern die Aufgabe, Beispiele für Projekte, die in Greenfoot erstellt wurden, im Internet anzusehen und zu finden. Die Schüler sind auf der Suche nach Projekten, während die Lehrer ihrer Arbeit nachgehen.

Der Lehrer erklärt den Schülern, wie sie Beispiele für abgeschlossene Projekte finden können, die in der Greenfoot-Umgebung erstellt wurden. So können beispielsweise in der Google-Suchmaschine Projekte anhand der Keywords *"Greenfoot Java project example"* oder ähnliches gefunden werden.

Der Lehrer erklärt den Schülern, dass es zwei Arten von Greenfoot-Projekten gibt:

1. eine, die in einem Webbrowser ausgeführt werden kann, und
2. eine andere, die heruntergeladen und dann in der Greenfoot-Umgebung geöffnet und ausgeführt werden kann.

Die Lehrkraft stellt Projekte vor:

1. Projekte, die direkt in einem Webbrowser ausgeführt werden können
2. Projekte, die nach dem Download in Greenfoot ausgeführt werden können.

Der Lehrer kann einige Beispielprojekte herunterladen oder auf einige Projekte zugreifen, indem er auf die Links zugreift.

4. Teambildung und Projektzuweisung

Gegenstand: Einbinden der Teilnehmer mit einfachen Aufgabenaufgaben in das Lernen auf Basis eines Projekts.

Zu besprechende Konzepte: -

Aktivität: Die Lehrkraft stellt Teams zusammen, bereitet eine Aufgabe vor und legt ein Projekt fest, an dem die Schülerinnen und Schüler arbeiten sollen. Einige Beispiele für die Aufgabe finden Sie auf Moodle. Die Lehrkraft wählt ein repräsentatives und wenig komplexes Projekt aus und definiert für dieses Projekt eine Aufgabe (Problem), die die Schülerinnen und Schüler lösen sollen.

5. Zusammenarbeit und Programmierung im Team

Gegenstand: Den Schülern wird erklärt, wie ein Greenfoot-Projekt zu erstellen ist. Die Schüler arbeiten an der zugewiesenen Aufgabe.

Zu besprechende Konzepte: Erstellen eines Greenfoot-Projekts

Aktivität: Der Lehrer startet Greenfoot und zeigt den Schülern, wie man ein Projekt erstellt. Die Lehrkraft weist den Schülerinnen und Schülern die Aufgabe zu, ein Greenfoot-Projekt zu erstellen.

Aufgabe (Task 1.1): Erstellen eines neuen Projekts, Vergabe eines Namens (z. B. **TowerDefense, wie Turmverteidigung**) und Speicherung an einem geeigneten Speicherort.

Der Lehrer erklärt den Schülern, dass das Projekt auf unterschiedliche Weise beschafft werden kann:

Möglichkeit 1: Laden von GIT[1] , indem Sie den folgenden Befehl eingeben

Commit: 9046f5353d857dcc112abd92d7b7170abcc64a80
https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/9046f5353d857dcc112abd92d7b7170abcc64a80

Möglichkeit 2: Download von git, aber ZIP-Datei von der folgenden Adresse

https://oop4fun.fon.bg.ac.rs/

Die Lehrkraft erklärt den Schülerinnen und Schülern, dass sie in der heutigen Unterrichtseinheit nicht weiter an dem gerade erstellten Projekt arbeiten werden. sondern Ab der nächsten Unterrichtsstunde soll an den bestehenden Projekten weitergearbeitet werden.

Die Schülerinnen und Schüler übernehmen die Aufgabe und lösen das Problem.

6. Peer-Review und Feedback

Gegenstand: Diskussion über die von den Schülern vorgeschlagene Lösung.

Zu besprechende Konzepte: -

Aktivität: Die Lehrkraft überwacht die Arbeit der Schülerinnen und Schüler und gibt ihnen bei Bedarf Lösungshinweise. Nach dem Ende wählt der Lehrer ein Team aus, das seine Lösung zeigt. Die Lehrkraft bespricht den Lösungsvorschlag mit den Schülerinnen und Schülern.

7. Hausaufgabe

Gegenstand: Ausgabe von Aufgaben an die Schüler, um die Greenfoot-Projekte zu erkunden.

Zu besprechende Konzepte: -

Aktivität: Die Lehrkraft definiert die Aufgabe, die die Schülerinnen und Schüler an einem der bestehenden Projekte lösen sollen.

8. Bewertung und Einstufung

Gegenstand: Einbeziehen der Schüler in den Bewertungsprozess der Projekte.

Zu besprechende Konzepte: -

Aktivität: Die Schüler stellen ihr Projekt vor. Die Lehrkraft fordert alle auf, sich an der Bewertung der vorgestellten Projekte zu beteiligen, indem sie den Schülerinnen und Schülern Links zum Ausfüllen eines Umfrage-Formulars sendet. Die Schülerinnen und Schüler bestimmen durch ein Punktesystem die drei besten Teams.

[1]Die Schüler müssen GIT auf ihrem Computer installiert haben.

Nach den Präsentationen gibt die Lehrkraft eine Zusammenfassung der Schülerprojekte. Der Lehrer teilt seine Eindrücke über die Arbeit der Schüler mit und gibt an, ob er damit zufrieden ist, ob sie seine Erwartungen erfüllt oder übertroffen haben.

3. Klassen und Objekte

Innerhalb der thematischen Einheit Klassendefinition und Objekte wurden zwei Unterrichtsszenarien erstellt.

3.1. Kennenlernen von Klassen und Objekten durch Spieleentwicklung mit Greenfoot

3.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 3. Kennenlernen von Klassen und Objekten durch Spielentwicklung mit Greenfoot

Titel	Kennenlernen von Klassen und Objekten durch Spielentwicklung mit Greenfoot
Lernziele	Am Ende dieser Unterrichtseinheit werden die Schüler in der Lage sein, **das grundlegende Konzept des Objekts und der Klasse zu verstehen.** Das Verständnis der untersuchten Konzepte wird im Rahmen der Spielentwicklung diskutiert und Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren mit dem Greenfoot-Tool gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse gefordert.
Dauer des Szenarios	1. Objekt (10 Minuten) 2. Identifizierung von Objekten und deren Eigenschaften (15 Minuten) 3. Klasse, Instanz (15 Minuten) 4. Orientierung in Greenfoot: World, Actor, MyWorld (10 Minuten) 5. Klassenkonstruktor (10 Minuten) 6. Aufgabe Task 1.2 (15 Minuten) 7. Hintergrundbild (10 Minuten) 8. Aufgabe Task 1.3 (15 Minuten)
Materialien und Ressourcen	Greenfoot-Webseite und Download-Anweisungen. Beispiele, die vom Lehrer vorbereitet wurden. Internetressourcen zur Identifizierung anderer Beispiele.
Beschreibung	In diesem 90-minütigen Unterrichtsszenario werden Schülerinnen und Schüler der Sekundarstufe durch **den Blickwinkel der Spieleentwicklung mit dem Greenfoot-Tool** in die Programmierprinzipien mit Klassen und Objekten eingeführt. Der Unterricht beginnt mit einer **10-minütigen Einführung** durch den Lehrer, die die Schüler in die Welt der objektorientierten Programmierung einführt, indem er das Konzept eines Objekts und seine Eigenschaften im wirklichen Leben erklärt. Danach gibt der Lehrer den Schülern eine Rush-Hour-Herausforderung (10 Minuten), die die Objekte und ihre Eigenschaften anhand der textuellen Beschreibung der Aufgabe identifizieren müssen. Danach erarbeitet die Lehrkraft gemeinsam mit den Schülerinnen und Schülern eine Lösung für die Aufgabe (**5**

	Minuten). Im weiteren Verlauf der Lektion erklärt der Lehrer den Unterschied zwischen der Klasse und dem Objekt. Auf der höchsten Abstraktionsebene wird das Konzept der Vererbung (**15 Minuten**) erläutert. Das Tutorial startet die Greenfoot-Umgebung und erläutert die Klassen Word, Actor und MyWord **(10 Minuten).** Die Lehrkraft erklärt und präsentiert den Quellcode, der generiert wurde und steht nach dem Erstellen der Projekte (10 Minuten) hinter den Klassen World, Actor, MyWorld. Der Lehrer beginnt mit Aufgabe 1.2 und zeigt den Schülern, wie sie eine Welt für die Anwendung erstellen **(10 Minuten).** Die Lehrkraft erklärt, wie man ein Bild für eine bestimmte Klasse einrichtet und arbeitet gemeinsam mit den Schülerinnen und Schülern an Task 1.3 **(20 Minuten).**
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird Github/Gitlab oder ein Learning-Management-System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem ihnen zur Verfügung gestellten Forum über das Lernmanagement-Tool fortsetzen.

3.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Objektkonzept

Gegenstand: Einführung der Schülerinnen und Schüler in das Objektkonzept anhand von Beispielen aus der Praxis.

Zu besprechende Konzepte: Objekt und Objekteigenschaften.

Aktivität: Die Lehrkraft führt den Begriff Objekt ein**.** Der Lehrer sollte dieses Konzept für die Schüler anhand von Beispielen aus dem wirklichen Leben nachvollziehbarer machen. Zum Beispiel könnte der Lehrer die Schüler nach ihrem Namen, ihrer Größe, ihrem Geburtsdatum, ihrer Augenfarbe usw. fragen. Der Lehrer stellt diese Fragen, um die Schüler dazu anzuregen, darüber nachzudenken, wie sie sich voneinander unterscheiden, und sie auf eine spätere Frage vorzubereiten: Wie unterscheiden sich die Schüler voneinander?

Der Lehrer kommt zu dem Schluss, dass jeder von uns (Lehrer, Schüler, andere Personen) Eigenschaften hat, durch die wir uns voneinander unterscheiden, und betont, dass jeder von uns eigentlich ein “Objekt” ist. Der Lehrer erklärt die Objekt-Eigenschaften (Attribute) so, dass jeder von uns einen spezifischen Wert für jede Eigenschaft besitzt, und damit jedes Individuum von anderen unterscheidbar ist. Daher unterscheiden sich Objekte voneinander auf der Grundlage der Werte, die sie für ihre jeweiligen Eigenschaften besitzen.

2. Identifizierung von Objekten und deren Eigenschaften

Gegenstand: Identifizierung von Objekten und deren Eigenschaften, zusammen mit den Schülerinnen und Schülern.

Zu besprechende Konzepte: Objekte und ihre Eigenschaften

Aktivität: Der Lehrer gibt den Schülerinnen und Schülern die Aufgabe, anhand eines Textes Objekte und deren Eigenschaften zu identifizieren.

3. Klasse, Instanz (Objekt)

Gegenstand: Erwerb von Kenntnissen über Klassen und Instanzen, d.h. Objekte. Klarmachen des Unterschieds zwischen Klasse und Objekt.

Zu besprechende Konzepte: Klasse, Instanz

Aktivität: Um das Konzept von Klasse, Klasseninstanzen (Objekten) und Vererbung zu erklären, gibt der Lehrer Beispiele aus dem alltäglichen Leben.

Der Lehrer stellt den Schülern Fragen, um ihnen den Unterschied zwischen der Klasse und dem Objekt zu zeigen. Der Lehrer leitet eine Diskussion über erkannte Objekte und deren Zuordnung zu Klassen.

4. Orientierung in Greenfoot: World, Actor, MyWorld

Gegenstand: Einführung der World-Klasse in Greenfoot.

Zu besprechende Konzepte: Instanz der World-Klasse in Greenfoot

Aktivität: Der Lehrer startet die Greenfoot-Umgebung und erstellt ein einfaches Projekt. Der Lehrererklärt, wie die Schüler ein Projekt in Greenfoot erstellen können.

Der Lehrer weist darauf hin, dass jedes Projekt, das in der Greenfoot-Umgebung erstellt wurde, drei Klassen enthält: **Word**, **Actor** und **MyWord**.

Der Lehrer stellt die **MyWorld-Klasse** und ihre Rolle vor.

Der Lehrer betont, dass der Hintergrund jeder in Greenfoot erstellten Anwendung aus Zellen besteht, die eine einzelne Matrix darstellen. Der Lehrer zeigt, wie man die Hintergrundgröße (Matrixdimension) und die Größe jeder Matrixzelle definiert.

Der Lehrer erklärt, dass sich die Objekte, die auf dem Hintergrund erscheinen, in einer dieser Zellen befinden. Die Lehrkraft zeigt den Quellcode hinter jeder dieser Klassen.

5. Vorbereitung der Welt (Task 1.2)

Gegenstand: Beteiligung der Schüler an der Projektaufgabe.

Zu besprechende Konzepte: Welt in Greenfoot

Aktivität: Der Lehrer gibt eine Aufgabe an die Schüler aus.

Aufgabe 1.2: Erstellen einer Welt mit einer Größe von 10x10 Zellen. Jede Zelle sollte 75 Pixel groß sein.

Der Lehrer verfolgt die Arbeit der Schüler und bittet am Ende einen Schüler, seine Lösung zu zeigen und zu beschreiben.

Beschreibung der Lösung:

Bearbeiten Sie den Quellcode der Klasse MyWorld (doppelklicken Sie darauf), um eine Welt mit der Größe 10x10 Zellen zu erstellen. Jede Zelle sollte 75 Pixel groß sein.

Commit: a593cd4a92d0fa0db78275614c3e41a2e96b4e57

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/a593cd4a92d0fa0db78275614c3e41a2e96b4e57

6. Hintergrundbild

Gegenstand: Festlegen des Hintergrundbildes der Welt im Greenfoot-Projekt.

Zu besprechende Konzepte: Hintergrundbild für die World-Klasse.

Aktivität: Der Lehrer erklärt den Schülern, dass der Hintergrund der Greenfoot-Anwendung (oder -Welt) ein Bild sein kann. Der Lehrer erklärt den Schülern, dass der Hintergrund entweder ein einzelnes Bild sein kann, das die gesamte Weltfläche abdeckt, oder ein Bild, das den Abmessungen der Zelle entspricht.

Der Lehrer zeigt, wie man ein Bild in der MyWorld-Klasse festlegt. Die Schülerinnen und Schüler folgen den Anweisungen der Lehrkraft und arbeiten Schritt für Schritt zusammen.

7. Vorbereiten von Weltgrafiken

Gegenstand: Bekanntmachen mit dem Hintergrund-Bild der Weltklasse

Zu besprechende Konzepte: Hintergrundbild der Welt-Klasse

Aktivität: Der Lehrer gibt den Schülern die Aufgabe, Task 1.3 zu erledigen.

Aufgabe Task 1.3: Erstellung eines geeigneten Bildes für den Welthintergrund und dessen Einbindung. Der Lehrer erklärt den Schülern, was er von ihnen erwartet. Der Lehrer kann Beispiele, die bereits vorbereitet wurden, herunterladen und zeigen. Der Lehrer stellt den Schülerinnen und Schülern einen Link oder einen Git-Befehl zur Verfügung, um das erste Projekt herunterzuladen, das sie mit dieser Aufgabe (Funktion) aktualisieren müssen. Das erste Projekt kann aus dem Git-Repository heruntergeladen werden.

Die Schülerinnen und Schüler erledigen die Aufgabe allein oder in der Gruppe. Die Lehrkraft überwacht die Arbeit der Schüler.

Der Lehrer demonstriert die Lösung Schritt für Schritt, während die Schüler den Anweisungen folgen.

Beschreibung der Lösung:

- Finden oder Erstellen eines geeigneten Bildes für den Welthintergrund. Dafür kann man entweder vorbereitete Images verwenden (wählen Sie den Punkt Bild setzen... aus dem Kontextmenü der Klasse MyWorld) oder ein benutzerdefiniertes Bild

(kopieren Sie das Bild in den Unterordner images Ihres Projektordners und wählen Sie es auf die gleiche Weise wie zuvor beschrieben aus).

- Als Hintergrund kann ein einzelnes Bild dienen, das die gesamte Weltfläche abdeckt (berechnen Sie die benötigte Größe des Bildes in Bezug auf die Größe der Welt) oder ein kleineres, das wiederholt kachelartig angezeigt wird (verwenden Sie ein quadratisches Bild mit der Größe der Zelle).

Commit:1184980643db082cfdd6bde9984bceaddf010d49

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/1184980643db082cfdd6bde9984bceaddf010d49

3.2. Erstellen von Klassen und Objekten in Greenfoot

3.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 4. Erstellen von Klassen und Objekten in Greenfoot

Titel	Erstellen von Klassen und Objekten durch Spielentwicklung mit Greenfoot
Lernziele	Als Ergebnis dieser Unterrichtseinheit werden die Teilnehmer die grundlegenden Konzepte des Objekts, der Klasse, der Klasseneigenschaften und der Methoden verstanden haben.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse vermittelt, einschließlich Objekte, Klassen, Klasseneigenschaften und Methoden.
Dauer des Szenarios	1. Grundlegende Konzepte (25 Minuten) 2. Aufgabe Task 1.4 (10 Minuten) 3. Aufgabe Task 1.5 (30 Minuten) 4. Schnittstelle des Objekts (5 Minuten) 5. Botschaft und Methode (15 Minuten) 6. Aufgabe Task 1.6 (30 Minuten) 7. Wiederholung (15 Minuten)
Materialien und Ressourcen	Greenfoot-Webseite und Download-Anweisungen. Beispiele, die vom Lehrer vorbereitet wurden. Internetressourcen zur Identifizierung anderer Beispiele.
Beschreibung	In dieser Sitzung durchlaufen die Teilnehmer mehrere strukturierte Aufgaben, um ihr Verständnis für objektorientierte Programmierkonzepte zu vertiefen. Zunächst verbringen sie 25 Minuten damit, eine Klasse mit dem Namen Enemy zu erstellen, gefolgt von einer fokussierten 15-minütigen Aufgabe, um die Attribute und Methoden innerhalb dieser Klasse zu definieren. Als Nächstes nehmen sie sich 30 Minuten Zeit, um ein Objekt der Enemy-Klasse zu instanziieren, wobei sie ihr Wissen über die Objekterstellung und -initialisierung anwenden. Das Konzept der Benutzeroberfläche eines Objekts wird dann in 5 Minuten untersucht, wobei der Schwerpunkt auf der Definition der Operationen liegt, die ein Objekt ausführen kann. Anschließend vertiefen sich die Schülerinnen und Schüler in 15 Minuten in die Konzepte von Nachrichten und Methoden und lernen, wie Objekte durch Methodenaufruf kommunizieren. Die Unterichtseinheit widmet weitere 30 Minuten der praktischen Anwendung, in

	der die Studierenden Nachrichten an ihre instanziierte Enemy-Instanz senden und so ihr Verständnis des Objektverhaltens vertiefen. Schließlich werden in einer 15-minütigen theoretischen Wiederholung Schlüsselkonzepte wie Objekte, Klassen, Instanzen, interner Zustand, Identität, Botschaften und Methoden rekapituliert, um ein umfassendes Verständnis der Lernziele der Sitzung zu gewährleisten.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem ihnen zur Verfügung gestellten Forum über das Lernmanagement-Tool fortsetzen.

3.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Grundlegende Konzepte

Gegenstand: Erlangen von Kenntnissen über Klassen, ihre Eigenschaften und Objekte.

Zu besprechende Konzepte: Klassen, Eigenschaften und Objekte

Aktivität: Während der Einführung in den Unterricht geht der Lehrer mit den Schülern die zuvor behandelten Konzepte durch. Durch Diskussion erklärt der Lehrer die Konzepte der Klasse, die Eigenschaften der Klassen und die Objekte.

Der Lehrer weist den Schülern eine schriftliche Aufgabe zu, um bestimmte Klassen, ihre Eigenschaften, zugehörigen Objekte und entsprechenden Werte innerhalb eines bereitgestellten Textes zu identifizieren.

Der Lehrer wählt einen Schüler aus der Klasse aus, um seine Lösung zu präsentieren. Andere Schüler beteiligen sich, indem sie ihre Meinung äußern.

Dann wird Schritt für Schritt erklärt, wie man eine Klasse in der Greenfoot-Umgebung erstellt. Die Schülerinnen und Schüler folgen den Anweisungen der Lehrkraft und arbeiten Schritt für Schritt zusammen.

2. Aufgabe Task 1.4

Gegenstand: Einbeziehen der Schüler in die Arbeit an dem Projekt.

Zu besprechende Konzepte: Erstellen einer Klasse

Aktivität: Der Lehrer gibt den Schülern die in Task 1.4 enthaltene Aufgabe: Erstellen Sie eine Klasse Enemy (Feind) und fügen Sie ein entsprechendes Bild dafür ein.

Der Lehrer erklärt den Schülern, was er von ihnen erwartet. Die Lehrkraft kann das Abschlussprojekt, das bereits von der Lehrkraft vorbereitet wurde, herunterladen und zeigen. Der Lehrer stellt den Schülerinnen und Schülern einen Link oder einen Git-Befehl zur Verfügung, um das erste Projekt herunterzuladen, das sie in dieser Aufgabe aktualisieren müssen. Das Projekt für diese Aufgabe kann aus dem Git-Repository heruntergeladen werden.

Die Schülerinnen und Schüler erledigen die Aufgabe einzeln oder in der Gruppe. Die Lehrkraft überwacht die Arbeit der Schüler. Der Lehrer demonstriert die Lösung Schritt für Schritt, während die Schüler den Anweisungen folgen.

Beschreibung der Aufgabe:

Erschaffe eine Enemy-Klasse für einen Feind im Spiel. Der Feind bewegt sich auf die Krone (vgl. Kronjuwelen) des Spielers zu, um sie zu beschädigen und schließlich zu zerstören. Erstellen Sie eine neue Unterklasse der Klasse Actor (wählen Sie den Punkt Actor mit rechter Maustaste und wählen Sie "Neue Unterklasse"... aus dem Kontextmenü der Klasse Actor). Geben Sie ihm den gewünschten Namen (Enemy, oder Feind) und weisen Sie ein Bild zu. Der Lehrer sollte die Konvention erklären, dass der Klassenname mit Großbuchstaben beginnen sollte, mit Bezug auf die gesamte Java-Namenskonvention.

Commit: 4981400623729c3d112b54454b6e6151e18426bf

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/4981400623729c3d112b54454b6e6151e18426bf

3. Aufgabe Task 1.5

Gegenstand: Führen Sie den ZUstands des Objekts ein. Lehren Sie den Schülern, eine Instanz der Klasse im Greenfoot zu erstellen.

Zu besprechende Konzepte: Zustand des Objekts, der Instanz

Aktivität: Der Lehrer erklärt das Konzept des Zustands des Objekts anhand eines einfachen Beispiels. Zum Beispiel wird der Lehrer in einem bestimmten Abstand zur Haustür im Klassenzimmer positioniert. Dieser Abstand bestimmt ihre Position, wobei Schritte nach vorne oder hinten ihre Nähe zur Tür verändern. Daher kann die Position des Lehrers relativ zur Haustür durch eine Variable quantifiziert werden, die sich im Laufe der Zeit ändert. Die Position des Lehrers wird also durch die Entfernung von der Haustür definiert, was veranschaulicht, wie der Zustand eines Objekts – in diesem Fall des Lehrers – durch den Wert seines Attributs (Entfernung) zu einem bestimmten Zeitpunkt charakterisiert wird.

Der Lehrer beschreibt Aufgabe 1.5, wie man eine Instanz der Klasse Enemy erstellt. Die Lehrkraft öffnet die letzte Version des Projekts, während die Schülerinnen und Schüler den Anweisungen der Lehrkraft folgen und Schritt für Schritt zusammenarbeiten.

Der Lehrer erstellt eine Instanz der Klasse Enemy (mit der rechten Maustaste). Diese ist aus dem Kontextmenü der Klasse Enemy auszuwählen und die Instanz mit einem linken Mausklick an der gewünschten Position in die Welt setzen. Danach wird der interne Zustand des Objekts inspiziert (wählen Sie mit der rechten Maustaste einen Gegenstand aus, platzieren Sie ihn in der Welt und wählen Sie Inspect aus dem Kontextmenü der erstellten Instanz).

Der Lehrer bittet die Schüler, eine neue Instanz zu erstellen und sie in eine andere Position zu bringen, um die internen Zustände zweier erstellter Instanzen zu vergleichen.

4. Schnittstelle des Objekts

Gegenstand: Stellen Sie den Schülern die Schnittstelle als eine Reihe von Aktionen vor, die wir an einem Objekt ausführen können.

Zu besprechende Konzepte: Schnittstelle

Aktivität: Die Lehrkraft führt die Schülerinnen und Schüler anhand einfacher Beispiele in das Konzept einer Schnittstelle ein. Wenn wir zum Beispiel eine Person und die Aktivitäten beobachten, die sie während des Tages ausführt (z. B. aufwachen, frühstücken, zur Arbeit gehen), ohne uns mit bestimmten Details zu befassen – z. B. wie sie aufwacht (ob durch einen Wecker, ein Telefon oder einen Elternteil), was und wo sie frühstückt oder mit welchem Verkehrsmittel sie zur Arbeit fährt –, kann die Sammlung dieser Aktivitäten mit einer Schnittstelle verglichen werden. Eine Schnittstelle definiert, welche Aktionen Objekte einer bestimmten Klasse ausführen können, gibt aber nicht an, wie diese Aktionen ausgeführt oder ausgeführt werden. Wir definieren nicht, was er zum Frühstück isst, wie er zur Arbeit gegangen ist (zu Fuß, mit dem Auto oder mit dem Bus) oder wie er aufgewacht ist (ob ihn jemand gerufen hat oder die Uhr ihn geweckt hat).

5. Botschaft und Methode

Gegenstand: Führen Sie die Studierenden in das Methodenkonzept ein.

Zu besprechende Konzepte: Methoden einer Klasse

Aktivität: Der Lehrer stellt das Konzept der Methode vor. Um das Konzept einer Methode zu erklären, verbindet der Lehrer Eigenschaften (Merkmale, Attribute) mit den sich ändernden Werten für diese Eigenschaften.

Beispiel: Betrachten wir die Klasse Person und das Lebensalter, so steigt ihr Wert jedes Jahr um eins, und zwar konstant am selben Tag. Auf der anderen Seite, wenn wir uns Eigenschaften wie Größe und Gewicht ansehen, ändern sich diese Eigenschaften häufig – Menschen wachsen und ihr Gewicht schwankt.

Beispiel: Wenn wir die Bewegung einer Person von Punkt A nach Punkt B beobachten und diese Bewegung in Schritten beschreiben – wie z.B. einen Schritt nach vorne, eine Linksdrehung um 45 Grad, eine 8-stufige Vorwärtsbewegung, eine Rechtsdrehung um 30 Grad und 5 weitere Schritte nach vorne –, können diese einzelnen Handlungen zu dem zusammengefasst werden, was wir eine Methode nennen.

Der Lehrer zeigt den Schülern in Greenfoot, wie sie sich die Methoden einer Klasse ansehen können, die auf ausgehend von einem bestimmten Objekt aufgerufen werden können.

Der Lehrer zeigt die Methoden, die in der Actor-Klasse definiert sind. Es wird auch gezeigt, wie Methoden für ein Objekt aufgerufen werden.

6. Aufgabe Task 1.6

Gegenstand: Einführung in den Aufruf einer Methode für ein Objekt.

Zu besprechende Konzepte: Methodenaufruf

Aktivität: Der Lehrer beschreibt die Aufgabe 1.6. Diese besteht darin, dem Instanz-Objekt der Klasse Enemy eine Botschaft zu senden, bzw. eine Methode aufzurufen. Die Lehrkraft öffnet die letzte Version des Projekts, während die Schülerinnen und Schüler den Anweisungen der Lehrkraft folgen und Schritt für Schritt zusammenarbeiten.

Der Lehrer sendet Nachrichten an die Instanz der Klasse Enemy (Feind), so dass sie sich zur Position [12, 6] bewegt und nach unten zeigt (durch Rechtsklick auf das Objekt, wählen Sie von Actor geerbt und dann die Methode setLocation(int, int)). Beschreiben Sie den Schülern, was mit der Instanz passieren wird. Wie wurde der interne Zustand der jeweiligen Instanz beeinflusst?

Der Lehrer zeigt den Schülern, wie Methoden definiert werden. Er erstellt eine setPosition(int x, int y)-Methode, um den Actor auf bestimmte Koordinaten zu platzieren. Der Lehrer betont, dass diese Methode äquivalent zu setLocation(int x, int y) ist, und betont, wie wichtig es ist, zu überprüfen, ob eine Methode bereits existiert, bevor eine neue definiert wird, um Duplikate zu vermeiden. Es wird betont, dass Methodennamen ihren Zweck und ihre Funktionalität klar angeben sollten, damit sie allein anhand des Namens sofort verstanden werden können. Der Lehrer betont auch, dass Methodennamen prägnant sein sollten, und gibt Beispiele für gut definierte, schlecht definierte und falsch definierte Methoden. Methoden, die der Benutzer ausführen (aufrufen) kann, werden durch einen Rechtsklick auf ein Objekt sichtbar.

Die Lehrkraft beschreibt, wie die Methode definiert wird. Die Schülerinnen und Schüler folgen den Anweisungen der Lehrkraft und arbeiten Schritt für Schritt zusammen.

Der Lehrer bittet die Schüler, die Methode zu definieren, mit der der Akteur abgesenkt wird (Verringerung der y-Koordinate) und die Methode, mit der er angehoben wird (Erhöhung der y-Koordinate). Der Lehrer betreut die Schüler während sie an der Aufgabe arbeiten und gibt bei Bedarf Hinweise.

7. Wiederholung

Gegenstand: Fassen Sie das Konzept zusammen, das in der Unterrichtseinheit behandelt wurde.

Zu besprechende Konzepte: Objekt, Klasse, Instanz, interner Zustand, Identität, Nachricht, Methode.

Aktivität: Der Lehrer fasst das Konzept zusammen, das in der Unterrichtseinheit behandelt wurde.

4. Algorithmen

Innerhalb der Themeneinheit Algorithmus wurden zwei Unterrichtsszenarien erstellt.

4.1. Einführung in Algorithmen innerhalb der Greenfoot-Umgebung

4.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 5. Einführung in Algorithmen und algorithmisches Denken

Titel	Einführung in Algorithmen und algorithmisches Denken
Lernziele	Als Ergebnis dieser Unterrichtseinheit sollten die Schüler über ein solides Verständnis von Algorithmen und algorithmischem Denken verfügen, in der Entwicklung und Implementierung grundlegender Algorithmen geübt sein und in der Lage sein, algorithmische Konzepte anzuwenden, um eine Vielzahl von Problemen effektiv zu lösen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Sie sollen grundlegende Programmierkenntnisse erlangen, einschließlich Variablen, Funktionen, Iterations- und Auswahlkonzepte, sich Kenntnisse im logischen Denken aneignen und Fähigkeiten zum Problemlösen entwickeln. Die Schüler sollten bereits mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Einführung in einfache Algorithmen als schrittweise Abläufe (15 min) 2) Aufgabe: Task 2.1 (20 Minuten) 3) Algorithmen und deren Eigenschaften (15 Minuten) 4) Aufgabe: Task 2.2 (25 Minuten) 5) Erstellen von Algorithmen (15 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	In dieser Unterichtseinheit tauchen Schülerinnen und Schüler der Sekundarstufe in das Reich der Algorithmen und des algorithmischen Denkens ein. Es beginnt mit einer 15-minütigen Einführung, die darauf abzielt, die Schüler mit grundlegenden algorithmischen Konzepten vertraut zu machen und ihre Bedeutung bei der Problemlösung hervorzuheben. Nach der Einführung werden die Schülerinnen und Schüler eine 20-minütige Aufgabe lösen, in der sie einen einfachen Algorithmus schreiben, um ein bestimmtes Problem zu lösen. Diese praktische Aktivität ermöglicht es den

	Schülern, die zuvor vorgestellten Konzepte anzuwenden und ihre algorithmischen Fähigkeiten zu verbessern. Anschließend wird in einem 15-minütigen Teil über Algorithmen und ihre Eigenschaften diskutiert. Zu den behandelten Themen gehören Korrektheit, Effizienz und Skalierbarkeit, wobei die Bedeutung klarer und präziser Anweisungen beim Algorithmendesign hervorgehoben wird. Aufbauend auf ihrem Verständnis verbringen die Schülerinnen und Schüler die nächsten 25 Minuten damit, einen allgemeineren Algorithmus für ein etwas komplexes Problem zu entwickeln. Diese Aufgabe fordert die Schülerinnen und Schüler heraus, abstrakt und kritisch zu denken und algorithmische Prinzipien anzuwenden, um reale Szenarien zu bewältigen. Im abschließenden 15-minütigen Segment beschäftigen sich die Studierenden mit der Algorithmisierung, Analyse und Verfeinerung ihrer Algorithmen. Dabei geht es darum, Verbesserungspotenziale zu identifizieren, die Effizienz zu optimieren und die Robustheit der Algorithmen sicherzustellen. Während des gesamten Unterrichts arbeiten die Schüler einzeln oder in kleinen Gruppen, um die Zusammenarbeit und das Peer-Learning zu fördern. Durch die aktive Teilnahme am Schreiben und Analysieren von Algorithmen entwickeln die Studierenden Fähigkeiten zum kritischen Denken und zur Problemlösung. Am Ende verfügen die Schüler über ein tieferes Verständnis von Algorithmen und algorithmischem Denken und haben wesentliche Fähigkeiten erlangt, um komplexe Probleme systematisch und effektiv anzugehen.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

4.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Einführung in einfache Algorithmen als schrittweise Abläufe

Gegenstand: Einführung des Begriffs Algorithmus.

Zu besprechende Konzepte: Grundlagen von Algorithmen

Aktivität: Der Lehrer führt das Konzept des Algorithmus anhand von Beispielen aus dem wirklichen Leben ein. Zum Beispiel fragt der Lehrer die Schüler, was sie morgens vom Aufwachen bis zum Start der Schule machen. Der Lehrer fragt dann die Schüler, ob sie wissen, wie man eine Pizza oder ein warmes Sandwich macht, oder ob einer von ihnen ein Rezept für die Zubereitung eines Gerichts oder eines Kuchens kennt.

Der Lehrer verbindet den Prozess der Zubereitung einer Mahlzeit oder eines Kuchens mit der Erstellung eines Programms und betont, dass es genauso wie ein Rezept für die Zubereitung einer Mahlzeit auch ein "Rezept" für die Erstellung eines Programms gibt, das als Algorithmus bezeichnet wird.

Der Lehrer erklärt anschließend, dass ein Algorithmus eine Reihe von Schritten ist, die definieren, wie ein Programm ausgeführt wird.

Der Lehrer erklärt weiter, dass diese Reihe von Schritten nicht unbedingt immer nacheinander ausgeführt werden muss. Es kann einige Schritte im Algorithmus geben, die nur abhängig von einer bestimmten Bedingung ausgeführt werden können. Der Lehrer bittet die Schüler, ein Beispiel für einen solchen Fall zu geben (z. B. wenn wir einen Algorithmus für die Zubereitung eines warmen Sandwiches schreiben, wenn wir keine Zutat haben, z. B. Schinken, aber eine ähnliche Zutat, können wir sie ersetzen oder die fehlende Zutat kaufen).

Der Lehrer erklärt den Schülern, dass einige Schritte im Algorithmus mehrmals wiederholt werden können. Die Lehrkraft bittet die Schülerinnen und Schüler, ein Beispiel für einen Algorithmus zu nennen, bei dem Schritte mehrmals wiederholt werden.

2. Aufgabe: Task 2.1

Gegenstand: Einführung in das Formulieren einfacher Algorithmen.

Zu besprechende Konzepte: Algorithmus

Aktivität: Die Lehrkraft gibt den Schülerinnen und Schülern die Aufgabe Task 2.1, das Verfahren auf Papier zu schreiben, mit dem sie beschreiben, wie ein Fußgänger die Straße überquert.

Zu Beginn gibt der Lehrer den Schülern keine zusätzlichen Anweisungen, sondern beobachtet nur, wie sie denken und arbeiten. Stellt jemand eine Frage, die für die Beschreibung der Anweisungen für Fußgänger, die die Straße überqueren, wichtig ist, lobt er ihn und betont, warum diese Information wichtig ist.

Nach einiger Zeit fragt der Lehrer die Schüler, ob sie darauf geachtet haben, wo der Fußgänger die Straße überquert, ob es sich um eine markierte Stelle zum Überqueren handelt oder nicht.

Außerdem fragt die Lehrkraft die Schülerinnen und Schüler, ob sie sich überlegt haben, ob es am Fußgängerüberweg eine Ampel gibt oder nicht.

Am Ende wählt der Lehrer einige Schüler aus, die ihre Anweisungen zum Überqueren der Straße vorlesen sollen.

3. Algorithmen und deren Eigeschaften

Gegenstand: Einführung in die Eigenschaften von Algorithmen

Zu besprechende Konzepte: Eigenschaften von Algorithmen

Aktivität: Der Lehrer erklärt den Schülern die Eigenschaften von Algorithmen. Die Lehrkraft erklärt den Schülerinnen und Schülern, dass Algorithmen auch grafisch dargestellt werden können und gibt ein Beispiel für Algorithmen, die grafisch dargestellt werden.

4. Aufgabe: Task 2.2.

Gegenstand: Schreiben eines allgemeineren Algorithmus

Zu besprechende Konzepte: Schreiben von Algorithmen

Aktivität: Es soll ein allgemeiner Algorithmus für die Zubereitung eines Heißgetränks geschrieben werden. Dabei ist zu überlegen, wie die Eingaben eines solchen Algorithmus aussehen müssen, damit er allgemein ist.

5. Erstellen von Algorithmen

Gegenstand: Geben Sie den Schülern mehr Beispiele und beziehen Sie die Schüler ein, um eigene Algorithmen zu definieren.

Zu besprechende Konzepte: Algorithmus

Aktivität: Die Lehrkraft erklärt den Schülern, dass es auch in der Mathematik bestimmte Algorithmen gibt, die wir zur Lösung von Problemen verwenden. Die Lehrkraft fragt die Schülerinnen und Schüler, ob sie ein Beispiel nennen können.

Das Beispiel, das der Lehrer den Schülern erklärt, ist die Berechnung des Wertes eines arithmetischen Ausdrucks, der mehrere mathematische Operationen hat, bei denen es notwendig ist, die Prioritätsregeln für die Ausführung mathematischer Operationen zu beachten.

Der Lehrer gibt auch einige andere Beispiele, wie z.B. das Zusammenbauen neuer Möbel, die gekauft wurden, und die Lieferung mit Anweisungen, die beschreiben, wie diese Möbel zusammenzubauen sind. Ein anderes Beispiel können die Anweisungen sein, die wir über das Navigationssystem erhalten, wenn wir über die Navigation von Punkt A nach Punkt B gelangen möchten.

Der Lehrer bittet die Schüler, ihren eigenen Algorithmus auf das Papier zu schreiben, woraufhin einige von ihnen das Ergebnis präsentieren.

4.2. Greenfoot-Adventure: Verstehen von Methodenaufrufen, Dokumentation und Anwendungssteuerung

4.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 6. Verstehen von Methodenaufrufen, Dokumentation und Anwendungssteuerung

Titel	Greenfoot Adventures: Verständnis von Methodenaufrufen, der Dokumentation und der Anwendungskontrolle von Java-Methoden
Lernziele	Am Ende des Unterrichts sollten die Teilnehmer ein solides Verständnis des Java-Methodenaufrufs haben, wobei sie sich insbesondere auf die Methoden 'act()' und 'move()' innerhalb der Greenfoot-Umgebung konzentrieren. Sie sollten in der Lage sein, das Schlüsselwort 'this' sinnvoll zu verwenden, um aktuelle Objekte innerhalb eines Klassenkontexts zu referenzieren. Darüber hinaus sollten die Teilnehmer die Fähigkeit demonstrieren, Methoden innerhalb einer Klasse aufzurufen und die Syntax und die Parameter zu verstehen, die für den Methodenaufruf erforderlich sind. Sie sollten auch Kenntnisse in der Anwendung von Methodenaufruftechniken nachweisen, um interaktive Spiel-entwicklungsaufgaben effektiv zu lösen. Darüber hinaus sollten die Schüler die Bedeutung der Codedokumentation verstehen und in der Lage sein, Java-Code effektiv zu dokumentieren, um Klarheit und Lesbarkeit zu gewährleisten. Schließlich sollten sie die Anwendungs-steuerungstechniken innerhalb von Greenfoot beherrschen, die eine präzise Manipulation und Interaktion mit Spielelementen ermöglichen, um ansprechende und funktionale Spielmechaniken zu schaffen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Als Vorwissen sind grundlegende Programmierkenntnisse, einschließlich Iterations- und Auswahlkonzepte vorteilhaft. Die Schüler sollten schon mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Erläuterung der Methode act () (10 Minuten) 2. Erklärung der Methode move() (20 Minuten) 3. Schlüsselwort this (5 Minuten) 4. Aufgabe Task 2.3 (10 Minuten) 5. Erklärung der Autovervollständigung (5 Minuten) 6. Die Bedeutung der Code-Dokumentation (15 Minuten) 7. Aufgabe Task 2.4 (5 Minuten) 8. Aufgabe Task 2.5 (5 Minuten) 9. Aufgabe Task 2.6 (10 Minuten) 10. Aufgabe Task 2.7 (20 Minuten) 11. Diskussion: Algorithmus, Eigenschaften, Algorithmisierung, Greenfoot-Buttons (5 Minuten)

Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab.Internet-Ressourcen.
Beschreibung	In diesem 105-minütigen Unterrichtsszenario begeben sich Schülerinnen und Schüler der Sekundarstufe auf eine Reise, um den Aufruf von Java-Methoden, die Codedokumentation und die Anwendungssteuerung in der Greenfoot-Umgebung zu beherrschen. Die Unterrichtseinheit beginnt mit einer 15-minütigen Erkundung der 'act()'-Methode, gefolgt von einem 10-minütigen Eintauchen in die 'move()'-Methode, wesentliche Bestandteile der Greenfoot-Programmierung. Die Schülerinnen und Schüler verbringen dann 5 Minuten damit, sich mit der Bedeutung des Schlüsselworts "this" bei der Referenzierung aktueller Objekte in einem Klassenkontext zu befassen. Im Anschluss daran erwartet die Schülerinnen und Schüler eine 10-minütige Aufgabe, in der sie das Aufrufen von Methoden innerhalb einer Klasse üben und die für den Methodenaufruf erforderlichen Syntax und Parameter anwenden müssen. Es folgt eine 5-minütige Erläuterung der Autovervollständigungs-funktionen (auto completion) in der Greenfoot-Umgebung, die zur Steigerung der Kodierungseffizienz beitragen kann. In den nächsten 15 Minuten werden die Teilnehmer die Bedeutung der Dokumentation der Programme verstehen und sehen, wie eine klare und prägnante Dokumentation die Lesbarkeit und Wartbarkeit des Codes verbessert. In einer 5-minütigen Aufgabe fügen die Schülerinnen und Schüler ihrem Code eine Dokumentation hinzu, die für Klarheit und Verständlichkeit für sich und andere sorgt. Aufbauend auf dieser Aufgabe verbringen die Teilnehmer weitere 5 Minuten damit, ihrem Code eine detailliertere Dokumentation hinzuzufügen. In einer 10-minütigen Aufgabe erkunden und lesen die Schülerinnen und Schüler Dokumentationen, die von ihren Kommilitonen hinzugefügt wurden, und erhalten Einblicke in verschiedene Codierungsstile und -ansätze. Schließlich verbringen die Schüler 20 Minuten damit, die Anwendungssteuerungstechniken in Greenfoot zu erkunden, die eine präzise Manipulation und Interaktion mit Spielelementen ermöglichen, um ansprechende und funktionale Spielmechaniken zu schaffen. Während des gesamten Unterrichts arbeiten die Schüler einzeln oder in kleinen Gruppen, um die Zusammenarbeit und das Peer-Learning zu fördern. Durch die aktive Teilnahme am Schreiben und Analysieren von Code entwickeln die Schüler Fähigkeiten zum kritischen Denken und zur Lösung von Problemen. Am Ende der Unterrichtseinheit erhalten die Teilnehmer ein tieferes Verständnis für den Aufruf von Java-Methoden, die Codedokumentation und die Anwendungssteuerung in Greenfoot und werden mit grundlegenden Fähigkeiten

	für die Spieleentwicklung und darüber hinaus ausgestattet.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und eines Lernmanagement- Systems (z.B. Moodle) verwendet. Die Studierenden können die Diskussion zum Thema in dem ihnen zur Verfügung gestellten Forum über das Lernmanagement-Tool fortsetzen.

4.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Erklärung der Methode act()

Gegenstand: Verwendung der act()-Methode. Es wird der Aufruf der Methode aus einer anderen Methode erklärt.

Zu besprechende Konzepte: Methode act()

Aktivität: Der Lehrer öffnet die letzte Version des TowerDefense-Projekts. Der Lehrer setzt Instanzen der Enemy-Klasse auf Position 0,0. Die Schüler werden gefragt, was passieren wird, wenn wir die act()-Methode für eine Instanz der Enemy-Klasse aufrufen. Ist es das erwartete Verhalten?

Der Lehrer bittet den Schüler, bei der zu erledigenden Aufgabe mitzuhelfen. Das Objekt der Enemy-Klasse ist zwei Zellen in die aktuelle Richtung zu bewegen, wenn die Methode act() augefrufen wird.

Der Lehrer zeigt den Schülern, wie sie diese Aufgabe erledigen sollen (Hinzufügen des move(int)-Methodenaufrufs innerhalb der act()-Methode), während die Schüler den Anweisungen des Lehrers folgen und ihre Ideen beisteuern sollen.

2. Erklärung der Methode move()

Gegenstand: Erklären der move()-Methode um ein Objekt vorwärts und rückwärts zu bewegen.

Zu besprechende Konzepte: Methoden für Bewegungen

Aktivität: Der Lehrer erklärt die Methode move(int). Der Lehrer stellt Instanzen der Enemy-Klasse an eine andere Position und ruft die Methode auf, indem er die positiven Werte übergibt, z.B. 1 oder 3. Der Lehrer fragt die Schüler, was sie denken, was passieren wird, wenn sie die Methode move() aufrufen und negative Werte eingeben, z.B. -1.

Der Lehrer weist den Schülern die Aufgabe zu, eine backward()-Methode zu schreiben, die ein Objekt um einen Schritt rückwärts verschiebt. Ist es möglich, das Objekt zwei oder mehr Schritte zurück zu verschieben? Was müssen wir tun?

3. Einführung des Schlüsselworts this

Gegenstand: Einführen des Schlüsselworts this

Zu besprechende Konzepte: this-Keword (Schüsselwort)

Aktivität: Der Lehrer erklärt das Schlüsselwort this. Mit this wird im Code eine Referenz auf das aufrufende Objekt erzeugt. Davon ausgehend, können objekteigene Attribute benutzt werden, und auch Methoden im Kontext des Objekts aufgerufen werden.

4. Aufgabe Task 2.3.

Gegenstand: Lehren Sie den Schülern, wie man eine Methode schreibt, um ein Objekt vertikal zu verschieben und den Positionswert der y-Koordinate zu ändern.

Zu besprechende Konzepte: Methodenaufruf

Aktivität: Der Lehrer fragt die Schüler, wie man das Objekt vertikal, nach oben oder unten bewegt. Der Lehrer bittet den Schüler, eine geeignete Methode für die Auf- und Abwärtsbewegung zu finden, indem er mit der rechten Maustaste auf ein Objekt klickt und die von actor geerbten Methoden auswählt. Die Schüler sollten folgende Methoden erkennen: turn, setRotation, setLocation, getLocation, getRotation. Der Lehrer weist den Schülern die Aufgabe zu, die Methode up() und down() zu schreiben. Die Lehrkraft überwacht die Arbeit der Schülerinnen und Schüler und erklärt am Ende gemeinsam mit ihnen, wie diese Methoden umgesetzt werden können. Die Methoden up() und down() ändern den Wert der Koordinate y, d.h. erhöhen bzw. verringern den Wert. Hier kann der Lehrer die Methodenparameter vorstellen, jedoch noch auf Details einzugehen.

5. Erklärung der Autovervollständigung

Gegenstand: Autovervollständigung in der Geeenfoot-Umgebung

Zu besprechende Konzepte: Funktion zur Code-Autovervollständigung (CRTL+SPACE)

Aktivität: Der Lehrer zeigt den Schülern, wie sie Methoden eines Objekts finden, die sie aufrufen können. Das ist dann zweckmäßig, wenn man den Namen einer Methode vergessen hat oder zu programmieren anfängt und die Greenfoot-Umgebung "bittet" zu helfen. Das entledigt den Programmierer von der Suche in der Dokumentation oder anderen Quellen und hilft, Code schneller zu schreiben.

6. Bedeutung der Codedokumentation

Gegenstand: Kommentare und Code-Dokumentation

Zu besprechende Konzepte: Kommentare und Dokumentation, Bekanntmachen mit dem Dokumentationsfenster.

Aktivität: Der Lehrer erklärt den Schülern, dass es Zeilen im Code gibt, die nicht Teil des auszuführenden Programms sind (Kommentare, Dokumentationskommentare).

Der Lehrer sollte die Unterschiede zwischen Kommentaren und Dokumentationen (als eine besondere Art von Kommentaren) betonen.

Der Lehrer präsentiert den Schülern den Quellcode der Enemy-Klasse und danach ein generiertes HTML-Dokument, das die Dokumentation der Enemy-Klasse beschreibt. Der Lehrer beschreibt hier, dass es eine Regel gibt, die wir befolgen müssen, wenn wir Dokumentation für unsere Klasse oder Methoden schreiben.

Der Lehrer weist den Schülern die Aufgabe zu, auszuprobieren, wie man Dokumentationen für Java-Klassen und -Methoden schreibt.

7. Aufgabe Task 2.4.

Gegenstand: Vorstellung des Methodendokumentations-Tags (Tag als Kennung)

Zu besprechende Konzepte: Tag für die Methodendokumentation

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, einen Dokumentationskommentar für die **Methode act()** hinzuzufügen .

Commit*:* 68b1c82c7df2c7826f2d3f78373498569adab7e9

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/68b1c82c7df2c7826f2d3f78373498569adab7e9

8. Aufgabe Task 2.5.

Gegenstand: Vorstellung des Klassendokumentations-Tags (Tag als Kennung)

Zu besprechende Konzepte: Tag für die Klassendokumentation

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, einen Dokumentationskommentar für die Methode act() hinzuzufügen.

Es soll der Dokumentationskommentar der Enemy-Klasse bearbeitet werden. Dazu soll die Version der Klasse und der Autor hinzugefügt werden. Die Änderungen auf der generierten HTLM-Seite können dann nachvollzogen werden.

Commit*:* 1a7a9f83c5271a7c0dfa46ce3b1ee65682b0c5e5

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/1a7a9f83c5271a7c0dfa46ce3b1ee65682b0c5e5

9. Aufgabe Task 2.6.

Gegenstand: Erkunden der Dokumentation der Klasse, um Methoden auszuwählen

Zu besprechende Konzepte: Umgang mit der Dokumentation

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, das Dokumentationsfenster anzusehen und die Dokumentation für die Actor- und Window-Klasse zu lesen. Der Lehrer betont, wie wichtig es ist, die Dokumentation zu lesen, um Methoden zu finden, die nützlich sein können.

10. Aufgabe Task 2.7.

Gegenstand: Vorstellung der Greenfoot-Bedienelemente (Buttons)

Zu besprechende Konzepte: Greenfoot-Bedienelemente (Buttons)

Aktivität: Der Lehrer arbeitet weiter an der letzten Version des Projekts. Der Lehrer bittet die Schüler, zwei Instanzobjekte der Enemy-Klasse hinzuzufügen und die act**()**-Methode für jede Instanz aufzurufen.

Der Lehrer beschreibt das Bedienelement "Greenfoot Act" und klickt darauf. Der Lehrer klickt auf den Act-Button, der sich im Hauptfenster befindet. Der Lehrer stellt eine Frage, zu erklären, was passiert ist.

Außerdem bittet der Lehrer den Schüler, auf den Button **"Run**" zu klicken und herauszufinden, was passiert ist.

Der Lehrer bittet die Schüler, auf die Schaltfläche “Reset” zu klicken und zu erklären, was passiert ist. Danach erklärt der Lehrer, was zu tun ist, damit jedes Mal, wenn der Reset-Knopf gedrückt wird, zwei Objekte der Enemy-Klasse an den Positionen (0,3) und (3,3) im Fenster erscheinen.

Der Lehrer sollte den Schülern erklären, dass es notwendig ist, den Konstruktor der World-Klasse so zu ändern, dass diese Objekte im Konstruktor erstellt und an den gewünschten Positionen platziert werden, damit die Objekte bei jedem Klick auf die Schaltfläche “Reset” erscheinen.

11. Diskussion: Algorithmus, Eigenschaften, Algorithmisierung, Greenfoot-Buttons

Gegenstand: Der Lehrer fasst die Unterichtseinheit zusammen.

Zu besprechende Konzepte: move-Methode zur Bewegungssteuerung, Greenfoot-Buttons, Dokumentation

Aktivität: Der Lehrer fasst die Lektion zusammen. Er kann kann hier die Wichtigkeit der Code-Dokumentation und Konventionen für die Bennennung der Methoden und Klassen betonen.

5. Programmverzeigungen

Im Rahmen der Themeneinheit Verzweigung wurden zwei Unterrichtsszenarien erstellt.

5.1. Anwenden von Programmverzeigungen in der Spieleentwicklung mit Greenfoot – Einseitige Verzweigungen

5.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 7. Einseitige Programmverzeigungen

Titel	Anwenden von Programmverzweigungen in der Spieleentwicklung mit Greenfoot – Einseitige Verzweigungen
Lernziele	In der Unterrichtseinheit werden einseitige Verzweigungen behandelt (zweiseitige und mehrfache Verzweigungen werden hier bewusst weggelassen). Die Interaktion der Actor-Objekte mit der Welt wird vorgestellt. Die Schüler sind in der Lage, Code unter Verwendung von Bedingungen zu schreiben. Nach Abschluss dieses Themas haben die Teilnehmer gelernt, wie sie einfache if-else-Verzweigungen verwenden und wie sie Entscheidungen in ihrem Code treffen, um das Verhalten ihres Spiels zu steuern. Sie erwerben Grundkenntnisse in der Programmier-sprache Java, lernen die notwendige Syntax und analysieren und verstehen Code, was ihnen hilft zu verstehen, warum sich ihr Spiel auf eine bestimmte Weise verhält und wie sie Probleme beheben können. Die Schüler sind in der Lage, ihre eigenen Spielprojekte zu erstellen, indem sie das Gelernte über objektorientierte Programmierung anwenden.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Die Schüler sollten bereits mit Greenfoot im Allgemeinen vertraut sein.
Dauer des Szenarios	1. Einführung (5 Minuten) 2. Erklärung des Codes (15 Minuten) 3. Einseitige Verzweigung (10 Minuten) 4. Beobachtung des Zustands der Actor-Objekte (10 Minuten) 5. Hinzufügen der Erkennung der Ränder der Spielfläche (10 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus GitHub/Gitlab und Internet-Ressourcen.
Beschreibung	Die Unterrichtseinheit beginnt mit einer 15-minütigen Code-Erklärung der turn()-Methode, die die Grundlage für das Verständnis einseitiger Code-Verzweigungen bildet. Diese Diskussion wird von der Lehrkraft geleitet und zielt darauf ab, die Schülerinnen und Schüler zu einem gemeinsamen Verständnis der Rolle der

	turn()-Methode der actor-Klasse zu bringen. Im Anschluss daran werden in einem 10-minütigen Abschnitt grundlegende Kenntnisse zu einseitigen Verzweigungen erworben. Diese Erwerbsphase ist entscheidend für die Schüler, um die Grundprinzipien zu verstehen, bevor sie sich mit dem Schreiben von Code befassen. Als nächstes nehmen die Schüler an einer 10-minütigen Untersuchung teil, bei der sie den Zustand des Spieler-Objekts in ihrem Code beobachten und so ein tieferes Verständnis des Programmablaufs auf der Grundlage des Quellcodes erlangen. In der anschließenden 10-minütigen Produktionsphase wird das Projekt um die Erkennung von Welträndern, das Hinzufügen von Verhalten und die Einrichtung des Spiels erweitert.
Bewertung	Diese Aktivität ermöglicht es den Lehrern, Bewertungsfeedback zu geben, das auf der Grundlage der Diskussionen und der Überwachung des Flipped Classroom gewonnen wird. Das Peer-Review-Assessment wird online im Rahmen einer Hausaufgabe durchgeführt. Dies erinnert die Schülerinnen und Schüler an wichtige Aspekte der Übung, veranlasst sie, die Arbeit anderer Schülerinnen und Schüler kritisch zu bewerten, gibt ihnen Einblicke in gute oder weniger gute Lösungen ihrer Mitschüler usw. und erhöht die Gesamtleistung der Lernergebnisse. Auch die Arbeit im Teamprojekt, an der die Schülerinnen und Schüler arbeiten, wird von diesen Lernergebnissen und Kenntnissen profitieren.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von GitHub/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

5.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Einführung

Gegenstand: Der Lehrer bespricht mit den Schülerinnen und Schülern die Konzepte, die in der vorherigen Lektion behandelt wurden. Der Lehrer stellt die Ziele für diese Unterrichtsstunde vor.

Zu besprechende Konzepte: Algorithmus, Code-Dokumentation

Aktivität: Im einleitenden Teil der Lektion wiederholt der Lehrer mit den Schülern die Konzepte, die er aus der vorherigen Lektion übernommen hat. Die Lehrkraft stellt dann den neuen Stoff vor, der in der heutigen Unterrichtseinheit gelernt wird. Um die heutige Lektion zu veranschaulichen, stellt die Lehrkraft ein Beispiel für einen Algorithmus mit einfacher Verzweigung vor. Ein Beispiel könnte der Algorithmus für das Überqueren der Straße an einem Fußgängerüberweg sein, an dem es keine Ampel gibt. Der Fußgänger überquert nicht sofort die Straße, sondern vergewissert sich zunächst, dass keine Fahrzeuge von links oder rechts kommen. Wenn keine Fahrzeuge kommen, dann überquert der Fußgänger die Straße.

2. Erklärung des Codes

Gegenstand:

Zu besprechende Konzepte:

Aktivität: Die Lehrkraft lädt die neueste Version des Projekts herunter:

- Von der Moodle-Plattform
- Aus dem GIT-Repository

Der Lehrer erstellt und platziert ein gegnerisches Klassenobjekt irgendwo auf dem Brett. Es werden einige Methoden der Actor-Klasse erläutert:

- move(int)
- turn(int)
- setRotation()

Während die Lehrkraft die Methoden erklärt, zeigt sie auch, wie bestimmte Eigenschaften der Klasse verändert werden (z.B. die Position des Objekts auf der Spielfläche, also die x- und y-Werte). Der Lehrer bespricht mit den Schülern, wie die act()-Methode so ergänzt werden kann, dass jedes Mal, wenn die act()-Methode aufgerufen wird, das Enemy-Objekt zwei Schritte vorwärts bewegt wird.

3. Einseitige Verzweigung

Gegenstand:

Zu besprechende Konzepte: Verzweigung, Einseitige Verzweigung

Aktivität: Die Lehrkraft arbeitet weiter an dem Projekt. Der Lehrer erzeugt ein Enemy-Objekt auf der Spielfläche. Der Lehrer erklärt den Schülern, wie sie überprüfen können, ob sich das Objekt in der oberen Hälfte der Spielfläche befindet, und wenn ja, die Meldung "Gefunden" anzeigen.

Der Lehrer erklärt dabei auch den Schülern die Methode showText(), die zur Anzeige von Text verwendet wird.

4. Beobachtung des Zustands der Spieler

Gegenstand:

Zu besprechende Konzepte: Interner Zustand der Objekte

Aktivität: Der Lehrer erstellt eine Instanz der Enemy-Klasse und platziert sie in der Mitte der Spielfläche. Der Lehrer öffnet ein Fenster mit dem internen Zustand des Objekts und positioniert es so, dass es sichtbar ist, während die Anwendung läuft. Dann kann die Anwendung ausgeführt werden, und beobachtet werden, wie sich die Werte der Attribute x**,** y und rotation in der Enemy-Klasse ändern, wenn verschiedene Methoden aufgerufen werden. Wie ändern sich diese Werte, wenn Sie sich bewegen (nach oben, unten, links und rechts) und sich drehen?

5. Hinzufügen der Erkennung der Ränder der Spielfäche (World-Objekt)

Gegenstand:

Zu besprechende Konzepte: Erkennung der Ränder der Spielfläche

Aktivität: Die Lehrkraft bespricht mit den Schülerinnen und Schülern, wie sie feststellen können, ob sich ein Objekt am Rand befindet oder nicht. Wenn wir die Abmessungen der Welt kennen, kann anhand der Position (x,y) bestimmt werden, ob sich ein Objekt am Rand befindet oder nicht.

Der Lehrer platziert eine Enemy-Instanz irgendwo auf der Spielfläche (aber nicht am Rand) und ruft die Methode *IsAtEdge()* auf. Er diskutiert mit den Schülern, was passiert ist. Der Lehrer bewegt den Feind an den Rand und beobachtet das Ergebnis der Methode *IsAtEdge(),* die nun true zurückgeben sollte.

Der Lehrer erklärt die Methode isAtEdge().

Task 3.2: Der Lehrer gibt den Schülern die Aufgabe, dem Körper der act()-Methode Code hinzuzufügen, um den Feind um 180° zu drehen, indem er die Eigenschaft setRotation() aufruft, wenn er den Rand der Welt erreicht.

Die Lehrkraft bespricht mit den Schülerinnen und Schülern, wie ein Objekt, das den Rand erreicht hat, seine Bewegung fortsetzen kann:

- rückwärts (ohne Drehen)
- rückwärts (mit Drehen)

Gemeinsam mit den Schülerinnen und Schülern vervollständigt die Lehrkraft den Programmcode der Methode act().

Führen Sie nun die act()-Methode aus und besprechen Sie, was mit dem Feind passiert, sobald er den Rand des World-Bereichs erreicht.

Commit*:* 4927c3ff7eb39b51ba2738f2ab500fd6c32e3bb4

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/4927c3ff7eb39b51ba2738f2ab500fd6c32e3bb4

5.2. Anwendung von Programmverzweigungen in der Spieleentwicklung mit Greenfoot – Zweiseitige Verzweigungen

5.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 8. Zweiseitige Programmverzweigungen

Titel	Anwenden von Programmverzweigungen in der Spieleentwicklung mit Greenfoot – Zweiseitige Verzweigungen
Lernziele	Diese Unterrichtseinheit behandelt einseitige und zweiseitige Verzweigungen (mehrfache Verzweigungen werden absichtlich weggelassen). Die Schüler werden in die Lage versetzt, Code unter Verwendung von Bedingungen zu schreiben. Nach Abschluss dieses Themas lernen die Teilnehmer, wie sie einfache if-else-Anweisungen verwenden und wie sie Entscheidungen in ihrem Code treffen, um das Verhalten ihres Spiels zu steuern. Sie erwerben Grundkenntnisse in der Programmiersprache Java, lernen die notwendige Syntax, analysieren und verstehen Code, was ihnen hilft zu verstehen, warum sich ihr Spiel auf eine bestimmte Weise verhält und wie sie Probleme beheben können. Die Schüler sind dann auch in der Lage, ihre eigenen Spielprojekte zu erstellen, indem sie das Gelernte über objektorientierte Programmierung anwenden.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Die Schüler sollten bereits mit Greenfoot im Allgemeinen vertraut sein.
Dauer des Szenarios	1) Hinzufügen der Klassen Direction und Orb (30 Minuten) 2) Erklärung der Kollissionserkennung (30 Minuten) 3) Aufgabe: Task 3.4 4) Aufgabe: Task 3.5 5) Aufgabe: Task 3.6 6) Erklärung des Programmcodes: Zweiseitige Verzweigung (15 Minuten) 7) Aufgabe: Task 3.7 (20 Minuten) 8) Aufgabe: Task 3.8 (30 Minuten) 9) Zusammenfassung (5 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus GitHub/Gitlab.Internet-Ressourcen.
Beschreibung	Die Unterrichtseinheit beginnt mit einer 30-minütigen Übung, in der die Java-Klassen Direction (Richtung) und Orb (Kugel) hinzugefügt werden. Dies hilft den Schülern, die objektorientierte Natur von Java und die Bedeutung der richtigen Strukturierung von Code zu verstehen. Es folgt eine 20-minütige Code-Erklärung, die sich auf die Kollisionserkennung konzentriert und den Schülern hilft,

	Objektinteraktionen in ihrer Spielumgebung zu verstehen. Dann verbringen die Schüler 10 Minuten damit, ihrem Projekt die Kollisionserkennung hinzuzufügen und dabei ihr Verständnis von Verzweigungen zu nutzen. Als nächstes werden Aufgaben von jeweils 15 Minuten zugewiesen, bei denen die Schüler feindliche Bewegungen sowohl in einem benutzerdefinierten als auch in einem anspruchsvolleren Setup vorhersagen sollen und so ihre Problemlösungs- und Analysefähigkeiten verbessern. Eine weitere 15-minütige Diskussionsrunde konzentriert sich auf die zweiseitige Verzweigung, um sicherzustellen, dass die Teilnehmer zwischen einseitigen und zweiseitigen Programmverzweigungen unterscheiden können. In einer 20-minütigen Aufgabe werden die Schüler in die Verwendung von zweiseitigen Verzweigungen für die Kollisionserkennung einbezogen, um ihr Gelerntes durch die Anwendung komplexer Konzepte in einer praktischen Umgebung zu festigen. Die Unterrichtseinheit wird mit einer herausfordernden 30-minütigen Untersuchung abgeschlossen, bei der die Schülerinnen und Schüler erneut die Bewegung des Gegners vorhersagen, diesmal mit den zusätzlichen Erfahrungen aus den vorherigen Aufgaben, um ihr kumuliertes Wissen anzuwenden. Am Ende ist eine 5-minütige Wiederholung der erlernten Konzepte vorgesehen.
Bewertung	Diese Aktivität ermöglicht es den Lehrern, auf der Grundlage der Diskussionen und der Überwachung des Flipped Classroom Bewertungsfeedback zu geben. Das Peer-Review-Assessment wird online im Rahmen einer Hausaufgabe durchgeführt. Dies erinnert die Schülerinnen und Schüler an wichtige Aspekte der Übung und veranlasst sie, die Arbeit anderer Schülerinnen und Schüler kritisch zu bewerten. Es gibt ihnen Einblicke in gute oder weniger gute Lösungen ihrer Mitschüler und erhöht die Gesamtleistung der Lernergebnisse. Auch die Arbeit im Teamprojekt, an der die Schülerinnen und Schüler arbeiten, wird von diesen Lernergebnissen und Kenntnissen profitieren.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von GitHub/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

5.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Hinzufügen der Klassen Direction und Orb

Gegenstand: Erlernen, wie dem Projekt eine neue Klasse hinzugefügt wird.

Zu besprechende Konzepte: Klasse

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, Task 3.3 zu erarbeiten. Der Lehrer verfolgt die Aktivitäten der Schüler und bittet am Ende einen Schüler, seine Arbeit zu präsentieren.

Task 3.3: Erstellen von zwei neuen Klassen, die von der Actor-Klasse abstammen. Die erste Klasse wird die Klasse Direction (Richtung) sein und die zweite Klasse Orb (Kugel). Dazu sind geeignete (max. 50x50 Pixel) Bilder in einem grafischen Editor vorzubereiten. Diese Bilder sind dann den neu erstellten Klassen zuzuordnen.

Commit*:* 4ed6b37e6d481181d8b340639aa03391406b6c2e

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/4ed6b37e6d481181d8b340639aa03391406b6c2e

2. Erklärung der Kollissionserkennung

Gegenstand: Erklärung der Kollisionserkennung.

Zu besprechende Konzepte: Kollisionserkennung

Aktivität: Der Lehrer erzeugt eine Instanz der Klasse **Enemy** auf dem Spielfeld und eine Instanz der Klasse **Direction** auf dem Spielfeld in der gleichen Zeile. Der Lehrer fügt der act()-Methode Code hinzu, so dass sich das Objekt einen Schritt vorwärts bewegt.

Die Lehrkraft erklärt den Schülerinnen und Schülern, wie sie feststellen können, ob sich zwei oder mehr Objekte auf dem Spielfeld an der gleichen Position (in derselben Zelle) befinden. Dafür wird die Methode isTouching() benutzt.

Der Lehrer und die Schüler ändern die act()-Methode der Enemy-Klasse (Feind), um zu erreichen, dass sich der Feind um 90° im Uhrzeigersinn dreht, wenn er sich gemeinsam mit einem Dirction-Objekt in derselben Zelle befindet.

Gemeinsam mit den Schülerinnen und Schülern beobachtet die Lehrkraft, was mit dem Rotationsattribut passiert.

3. Aufgabe: Task 3.4

Gegenstand: Feststellung, ob sich zwei Objekte in der selben Zelle befinden.

Zu besprechende Konzepte: -

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, Task 3.4 zu bearbeiten. Der Lehrer verfolgt die Aktivitäten der Schüler und bittet am Ende einen Schüler, seine Arbeit zu präsentieren. Ein ausgewählter Schüler beschreibt und präsentiert seine Arbeit.

Task 3.4: Fügen Sie der act()-Methode der Enemy-Klasse Code hinzu, um folgendes Verhalten herzustellen: Der Spieler dreht sich um 90° gegen den Uhrzeigersinn, wenn er eine Zelle betritt, in der sich eine Instanz der Klasse Orb befindet.

Commit*:* 968e6f195e3def25e11bc41b664ba1715f7da11d

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/968e6f195e3def25e11bc41b664ba1715f7da11d

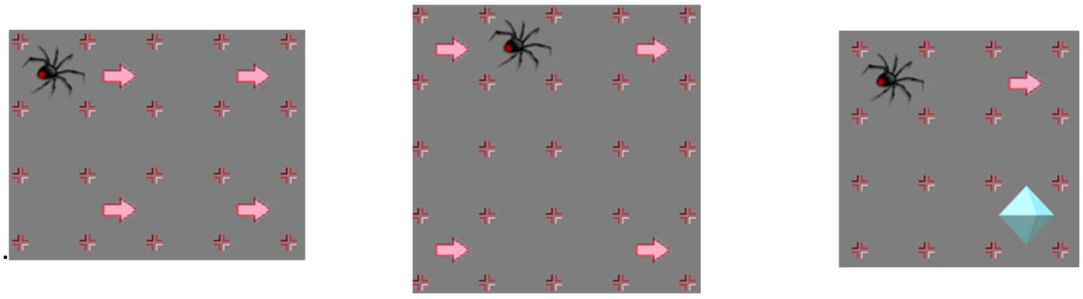
4. Aufgabe: Task 3.5

Gegenstand: Verständnis der Bewegung von Objekten.

Zu besprechende Konzepte: Bewegung von Objekten

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, Task 3.5 zu bearbeiten. Der Lehrer verfolgt die Aktivitäten der Schüler und bittet am Ende einen Schüler, seine Arbeit zu präsentieren. Dieser Schüler beschreibt und präsentiert seine Arbeit.

Tasks 3.5: Vorbereitung verschiedener Konfigurationen. Vorlagen finden Sie in den folgenden Abbildungen. Es soll greaten werden, wie sich der Feind bewegen wird? Führen Sie die Anwendung aus. Stimmt Ihre Vorhersage mit dem überein, was Sie beobachten? Was verursachte Unterschiede in der Vorhersage und der Realität?



a) b) c)

Abbildung 1 : Konfigurationen von benutzerdefinierten Setups von Instanzen zur Vorhersage der Bewegung von Enemy-Objekten. Die Enemy-Klasse wurde hier mit dem Bild einer Spinne verbunden.

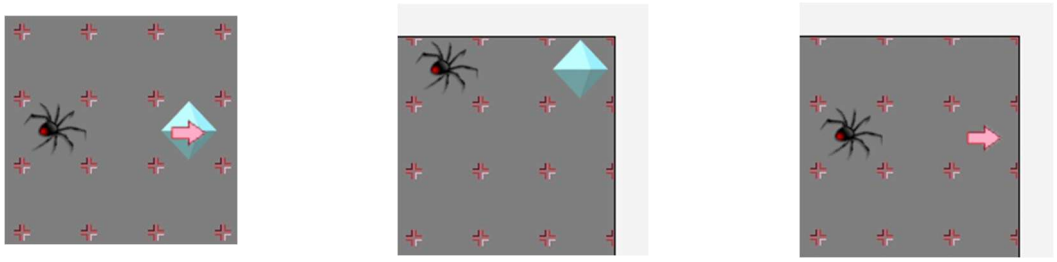
5. Aufgabe: Task 3.6

Gegenstand: Verstandene Bewegung von Objekten.

Zu besprechende Konzepte: Bewegung von Objekten

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, Task 3.6 zu bearbeiten. Der Lehrer verfolgt die Aktivitäten der Schüler und bittet am Ende einen Schüler, seine Arbeit zu präsentieren. Dieser Schüler beschreibt und präsentiert seine Arbeit.

Aufgabe 3.6: Bereiten Sie die Situation vor, so wie in der Abbildung unten dargestellt. Es soll vorhergesagt werden, wie sich der Feind bewegen wird. Führen Sie die Anwendung aus. Stimmt Ihre Vorhersage mit dem überein, was Sie beobachten? Was verursacht Unterschiede zwischen der Vorhersage und der Realität?



a) b) c)

Abbildung 2: Anspruchsvollere Objektplatzierungen zur Vorhersage der Bewegung von Instanzen der Klasse Enemy

6. Erklärung des Programmcodes: Zweiseitige Verzweigung

Gegenstand: Der Lehrer vermittelt den Schülern, wann und wie sie die Anweisung if-then-else und switch verwenden sollten.

Zu besprechende Konzepte: if-then-else-Anweisung, verschachtelte if-Anweisung, switch-Anweisung

Aktivität: Die Lehrkraft gibt den Schülerinnen und Schülern die Aufgabe, auf Papier zu beschreiben, wie ein Fußgänger die Straße überquert. Die Lehrkraft überwacht, wie die Schülerinnen und Schüler die Aufgabe lösen. Irgendwann legt der Lehrer den Schülern nahe, darauf zu achten, ob an der Stelle, an der die Straße überquert wird, eine Ampel steht oder nicht. Wenn in der Zwischenzeit, also bevor die Lehrer dies betonen, einer der Schüler diese Frage oder etwas Ähnliches stellt, lobt er ihn öffentlich und betont, dass es wichtig ist, dass wir vor dem Programmieren immer zuerst das Problem analysieren und alle Fälle identifizieren, in denen es vorkommen kann. Hier sollten die Schüler die vollständige und verschachtelte Verzweigung verstehen.

Der Lehrer wird die Schüler bitten, die Objekte der Orb- und Direction-Klasse an den linken, rechten oder rechten Rand zu platzieren – an die verschiedenen Grenzen des Spielfeldes. Die Schüler beschreiben das unpassende Verhalten des Feindes. Der Lehrer sollte erklären, dass es zwei Bedingungen gibt, die in einer Situation erfüllt sind. Der Feind berührt die Klasse Direction und berührt die Kante. Indem sie die Situation an der Tafel illustrieren, erkennen die Schüler, dass eine vollständige und verschachtelte Verzweigung erforderlich ist, und dann ändern sie den Code der Enemy-Klasse.

7. Aufgabe: Task 3.7

Gegenstand: Verständnis der zweiseitigen Verzweigungm, verschachtelten Verzweigungen und der Mehrfachverzweigung mit der switch-Anweisung.

Zu besprechende Konzepte: verschachtelte if-Anweisung, switch-Anweisung

Aktivität: Der Lehrer weist den Schülern die Aufgabe zu, Task 3.7 zu bearbeiten. Der Lehrer verfolgt die Aktivitäten der Schüler und bittet am Ende einen Schüler, seine Arbeit zu präsentieren. Dieser Schüler beschreibt und präsentiert seine Arbeit.

Task 3.7. Änderung des Codes der Methode act() der Klasse Enemy, damit eine mehrseitige Verzweigung verwendet wird. Dabei wird eine verschachtelte Bedingung verwendet. Erklären Sie die Erkennung der Kante zur primären Bedingung, überprüfen Sie danach die Berührung mit einem Direction-Objekt und überprüfen Sie dann schließlich die Berührung mit einem Orb-Objekt.

Commit*:* f017de8b49d4fc77f62afac4d842429560bcfb8b

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/f017de8b49d4fc77f62afac4d842429560bcfb8b

8. Aufgabe: Task 3.8

Gegenstand: Vorhersage der feindlichen Bewegung bei benutzerdefinierten Setups

Zu besprechende Konzepte: Verhalten von Objekten

Aktivität: Die Lehrkraft stellt willkürlich Objekte in das Spielfeld, und die Schülerinnen und Schüler erklären ihre Bewegung und ihr Verhalten (einzeln oder zu zweit).

Tasks 3.8: Führen Sie die Aufgaben Task 3.5 und Task 3.6 erneut aus.
Was hat sich geändert?

9. Zusammenfassung

Gegenstand: Der Lehrer fass die Unterrichtseinheit zusammen.

Zu besprechende Konzepte: Unterscheidliche Formen der Verzweigung im Code

Aktivität: Der Lehrer fass die Unterrichtseinheit zusammen.

6. Variable und Ausdrücke

Für das Gebiet Variable und Ausdrücke wurden fünf Unterrichtsszenarien erstellt.

6.1. Einführung in Variable und Datentypen im Greenfoot-Umfeld

6.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 9. Einführung in Variablen und Datentypen im Greenfoot-Umfeld

Titel	Einführung in Variablen und Datentypen im Greenfoot-Umfeld
Lernziele	Nach dieser Unterrichtseinheit werden die Teilnehmer in der Lage sein, Datentypen und Variablen anzuwenden. Das Verständnis der untersuchten Konzepte wird im Rahmen der Spieleentwicklung diskutiert, wodurch Kreativität, Teamwork und eine enthusiastische Arbeitsweise beim Programmieren mit Greenfoot gefördert werden.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden Grundlegende Programmierkenntnisse, einschließlich Variablen und Datentypen vermittelt. Die Schüler sollten bereits mit Greenfoot sein.
Dauer des Szenarios	1. Einführung (10 Minuten) 2. Identifizierung von Variablen (5 Minuten) 3. Datentypen (15 Minuten) 4. Deklaration von Variablen (10 Minuten) 5. Initialisierung von Variablen (5 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab.Internet-Ressourcen.
Beschreibung	In diesem 45-minütigen Unterrichtsszenario werden Schülerinnen und Schüler der Sekundarstufe mit dem Blickwinkel der Spieleentwicklung mit dem Greenfoot-Tool in die Verwendung von Datentypen und Variablen eingeführt. Der Unterricht beginnt mit einer 10-minütigen Einführung durch den Lehrer, die den Kontext zu den vorherigen Unterrichtseinheiten herstellt und die Grundlage für die Einführung und Definition der Variablen bildet.

	Darauf folgt ein 5-minütiger Block, in dem Schüler und Lehrer überlegen und Variablen für ihr Spiel identifizieren. Es wird herausgearbeitet, dass jede Variable einen bestimmten Typ haben muss. In diesem Kontext werden in den nächsten 15 Minuten verschiedene Datentypen präsentiert. Zu den Kernaktivitäten gehört eine 10-minütige Sitzung unter Anleitung des Lehrers, in der die Schüler Variablen für ihr Spiel deklarieren und dabei die Bedeutung von Variablennamen und -typen lernen. Darauf folgt ein 5-minütiger Abschnitt zur Variableninitialisierung, in der den Variablen Werte zugewiesen werden. In diesem Zusammenhang kann das Verhalten von Objekten im Spiel geändert werden (z.B. Drehung eines Objekts, Bewegung eines Objekts). Die Schülerinnen und Schüler werden die Arbeit an dem Spiel fortsetzen, das in den vorangegangenen Unterrichtseinheiten erklärt und begonnen wurde. Als Ergebnis werden am Ende der Sitzung neue Konzepte in Bezug auf Variablen und Datentypen eingeführt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

6.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Einführung

Im Einführungsteil wird der Kontext zu den vorangegangenen Unterrichtseinheiten hergestellt. Der Lehrer führt den Begriff *Variable ein*.

2. Identifizierung von Variablen

Gegenstand: Identifizierung von Variablen durch Diskussion, wobei die Rolle von Variablen in der Programmierung hervorgehoben wird.

Zu besprechende Konzepte: Datentypen, Variable und Werte.

Aktivität:

1. Der Lehrer führt den Begriff *Variable ein*,
2. Die Schüler können aufgefordert werden, Variablen für ihr Spiel herauszufinden und zu identifizieren,
3. Die Variablen können vom Lehrer und den Mitschülern diskutiert werden,
4. Vorerst kann der Variablentyp weggelassen werden (oder allgemein diskutiert werden).

3. Datentypen

Gegenstand: Verständnis des Konzepts von Datentypen, Erkennen der Anwendung von Variablen, Konzentriation auf die Variablentypen, die für die Spieleentwicklung benötigt werden.

Zu besprechende Konzepte: Variablen, Datentypen, Beispiele für reale Datentypen, Variablentypen für das Spiel.

Aktivität:

- Der Lehrer führt den Begriff *Datentyp ein*,
- Beispiele aus der Praxis können besprochen werden (z. B. ganze Zahlen für die Anzahl der derzeit anwesenden Schüler, Dezimalzahlen für einen Produktpreis, Text (String) für einen Instant-Messaging-Text usw.),
- Datentypen werden im Kontext der Greenfoot-Umgebung und der Programmiersprache Java betrachtet,
- Detaillierte Diskussion in Bezug auf Variablentypen, die für das Spiel erforderlich sind.

4. Deklaration von Variablen

Gegenstand: Anwenden des Wissens über Datentypen, Variablen und das Deklarieren von Variablen, insbesondere die für die Spieleentwicklung erforderlich sind.

Zu besprechende Konzepte: Variablen, Datentypen, Datentypen für das Spiel.

Aktivität:

- Datentypen werden im Kontext der Greenfoot-Umgebung und der Programmiersprache Java betrachtet,
- Der Lehrer sollte den Unterschied zwischen der Deklaration und der Initialisierung von Variablen erklären:
  - Bei der Variablendeklaration wird eine Variable eines bestimmten Datentyps deklariert, dabei kann schon ein Wert vorhanden sein, der Wert muss aber nicht vorhanden sein.
  - Es kann eine Analogie benutzt werden (z. B. beschriftetes Eingabefeld für die Telefonnummer, aber noch ohne Telefonnummer innerhalb des beschrifteten Eingabefeldes)
- Deklaration von Variablen, die für das Spiel erforderlich sind.
- Weitere Beispiele können erklärt werden. Wenn beispielsweise die Methode *act()* betrachtet wird, kann eine Variable für die Anzeige von Text deklariert werden.

5. Initialisierung von Variablen

Gegenstand: Benutzen von Datentypen und Variablen und das Initialisieren von Variablen, die für das Spiel erforderlich sind.

Zu besprechende Konzepte: Variablen, Datentypen, Variablenwerte für das Spiel.

Aktivität:

1. Auf Basis der zuvor vorgestellten Datentypen werden deren Datenwerte und Wertebereiche eingeführt,
2. Datenwerte und Wertebereiche werden im Kontext der Greenfoot-Umgebung und der Programmiersprache Java betrachtet,
3. Bei der Variableninitialisierung wird eine Variable eines bestimmten Datentyps deklariert und kann gleichzeitig initialisiert werden (dies wird als Anfangswert bezeichnet) und kann direkt im betrachteten Code geändert werden.
4. Initialisierung von Variablen, die für das Spiel erforderlich sind.
5. Weitere Beispiele können erklärt werden. Wenn beispielsweise die Methode act() betrachtet wird, kann eine Variable für die Anzeige von Text initialisiert werden.

6.2. Einführung in Operatoren und Ausdrücke

6.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 10. Einführung in Operatoren und Ausdrücke

Titel	Einführung in Operatoren und Ausdrücke
Lernziele	Nach dieser Unterrichtseinheit werden die Schüler in der Lage sein, Operatoren in programmiersprachlichen Ausdrücken anzuwenden. Dafür werden verschiedene Operatortypen (d. h. arithmetische Operatoren, boolesche Operatoren, relationale Operatoren) sowie die entsprechenden Ausdrücke vorgestellt. Darüber hinaus werden der Objektausdruck und die Referenzvariable eingeführt. Die untersuchten Konzepte werden für die Spieleentwicklung angewendet. Dabei sollen auch Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren gefördert werden.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Grundlegende Programmierkenntnisse, einschließlich Iteration und Programmverzweigungen werden vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Operatoren (15 Minuten) 2. Arithmetische Operatoren und Ausdrücke (10 Minuten) 3. Boolesche Operatoren (15 Minuten) 4. Relationale Operatoren (10 Minuten) 5. Boolesche Ausdrücke (10 Minuten) 6. Ausdrücke mit Objekten (5 Minuten) 7. Referenzvariable und ihr NULL-Wert (15 Minuten) 8. Aufgabe – Objekte in ihrer Ausrichtung drehen (15 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 95-minütigen Unterrichtseinheit werden Schülerinnen und Schüler der Sekundarstufe in Programmierkonzepte im Zusammenhang mit Operatoren und Ausdrücken eingeführt. Das passiert im Rahmen der Spieleentwicklung mit Greenfoot. Der Unterricht beginnt mit einer 15-minütigen Einführung durch den Lehrer, die Hinter-grundinformationen aus früheren Unterrichtseinheiten liefert, die die Grundlage für die Einführung und Definition eines Operators bilden. Darauf folgt ein 10-minütiger Abschnitt, in dem arithmetische Operatoren und

	Ausdrücke besprochen werden. Arithmetische Operatoren werden für arithmetische Operationen verwendet. In diesem Zusammenhang werden verschiedene Operatoren und Ausdrücke praxisnah erklärt und diskutiert. Im nächsten 15-minütigen Abschnitt werden boolesche Operatoren vorgestellt. Dabei handelt es sich um logische Operatoren, die zur Verknüpfung und Umwandlung boolescher Werte verwendet werden. Darauf folgt ein 10-minütiger Abschnitt, der sich auf relationale Operatoren bezieht, die zum Vergleichen von Werten verwendet werden. Basierend auf den vorangegangenen Sitzungen werden im nächsten 10-minütigen Szenario boolesche Ausdrücke erläutert. Der nächste 5-minütige Teil konzentriert sich auf den Objektausdruck, während im folgenden 15-minütigen Abschnitt Referenzvariablen behandelt werden. Schließlich folgt ein 15-minütiger Abschnitt unter Anleitung des Lehrers, in dem die Schüler Aufgaben lösen, die mit der Drehrichtung der Objekte im Spiel zusammenhängen. Dabei geben die Lehrkraft und die Mitschüler Feedback. Die Schülerinnen und Schüler werden das in den vorangegangenen Unterrichtseinheiten erstellte Spielprojekt fortsetzen und jetzt die Arbeiten vornehmen, die sich auf Operatoren und Ausdrücke beziehen.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Der Stand des Projekts eröffnet Möglichkeiten für Hausaufgaben. In diesem Zusammenhang können weitere Klassen, Ausdrücke und Werte eingeführt werden, um komplexeres Verhalten im Spiel zu erreichen (z.B. Teleports, Tunnel, etc.). Diese Möglichkeiten können mit den Schülern besprochen und die jeweilige Umsetzung als Hausaufgabe vergeben werden.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

6.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Operatoren

Gegenstand: Verständnis des Konzepts von Operatoren, auch in Bezug auf reale Szenarien, Kennenlernen verschiedener Arten von Operatoren.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für Operatoren aus der Praxis.

Aktivität: Der Lehrer führt den Begriff *Operator* ein. Der Lehrer sollte diesen Begriff den Schülern anhand von Beispielen aus dem wirklichen Leben (z. B. Einkauf von Produkten auf dem Markt) näher bringen. Anschließend stellt der Lehrer verschiedene Operatortypen vor.

2. Arithmetische Operatoren und Ausdrücke

Gegenstand: Verständnis der arithmetischen Operatoren, dabei soll sich auf reale Szenarien bezogen werden. Die Schüler lernen verschiedene arithmetische Operatoren kennen.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für arithmetische Operatoren aus der Praxis.

Aktivität: Die Lehrkraft kann Operatoren erklären, die bereits aus anderen Fächern bekannt sind (z.B. mathematische Arithmetikoperatoren). Diese Operatoren werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer diskutiert verschiedene Begriffe: Operator, Operand, Operatorrangfolge. Weitere Beispiele können folgen, beispielsweise kann das Definieren lokaler Variablen besprochen werden, um die x-Position und y-Position einer Entität abzurufen und zu manipulieren, wodurch ihre Position durch Erhöhen der Variablenwerte geändert wird.

3. Boolesche Operatoren

Gegenstand: Verständnis der booleschen Operatoren. Es soll sich auf reale Szenarien bezogen werden und verschiedene boolesche Operatoren vermittelt werden.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für boolesche Operatoren aus der Praxis.

Aktivität: Die Lehrkraft kann Operatoren erklären, die bereits aus anderen Fächern bekannt sind (z.B. Boolesche Operatoren aus der Mathematik). Diese Operatoren werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer diskutiert verschiedene Begriffe: Operator, Operand, Operatorrangfolge. Ein Beispiel ist die Verwendung eines booleschen Operators, um zu bestimmen, ob sich ein Objekt auf einer Diagonale befindet.

4. Relationale Operatoren

Gegenstand: Kennenlernen und Verständnis von relationalen Operatoren.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für relationale Operatoren aus der Praxis.

Aktivität: Die Lehrkraft kann Operatoren erklären, die bereits aus anderen Fächern bekannt sind (hier mathematische relationale Operatoren). Diese Operatoren werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer diskutiert verschiedene Begriffe: Operator, Operand, Rangfolge im Zusammenhang mit logischen Operatoren. Ein Beispiel kann das Benutzen lokaler Variablen sein, um zu überprüfen, ob die y-Position eines Objekts unter der eines anderen liegt. Relationale Operatoren werden typischerweise zur Bestimmung von Positionsbeziehungen zwischen Objekten benutzt.

5. Boolesche Ausdrücke

Gegenstand: Kennenlernen und Verständnis boolescher Ausdrücke, Bezug zu realen Szenarien.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für boolesche Ausdrücke aus der Praxis.

Aktivität: Der Lehrer erklärt Boolesche Ausdrücke im Kontext der zuvor vorgestellten Operatoren. Die Ausdrücke werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer erläutert die Rangfolge von Operatoren, Operanden innerhalb von booleschen Ausdrücken. Weitere Beispiele können folgen, beispielsweise die Verwendung Boolescher Ausdrücke um zu überprüfen, ob sich die Position eines Objekts innerhalb der definierten Arenadimension des Spielfelds befindet.

6. Ausdrücke mit Objekten

Gegenstand: Kennenlernen und Verstehen von Objektausdrücken. Bezug auf reale Szenarien.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für Objektausdrücke aus der Praxis.

Aktivität: Der Lehrer kann den Objektausdruck im Kontext des objektorientierten Designs erklären. Diese Ausdrücke werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer behandelt Operatoren, Operanden, Operatorrangfolge und Klassenumwandlung im Kontext von Objektausdrücken. Weitere Beispiele können folgen, beispielsweise das Vergleichen von Referenzen zweier Objekte, um zu überprüfen, ob sie sich überlappen.

7. Referenzvariable

Gegenstand: Verständnis von Referenzvariablen. Bezug und Anwendung auf reale Szenarien und Anwendung in der Spieleentwicklung.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Beispiele für Referenzvariablen aus der Praxis.

Aktivität: Der Lehrer erklärt Referenzvariablen im Kontext des objektorientierten Designs. Diese Referenzvariablen werden im Kontext der Programmiersprache Java betrachtet. Der Lehrer sollte dabei auch den Null-Referenzwert erklären.

8. Aufgabe – Objekte in ihrer Ausrichtung drehen

Gegenstand: Vertiefung des Verständnisses über Variablen, Datentypen, Operatoren und Ausdrücke und deren Verwendung in der Spielentwicklung.

Zu besprechende Konzepte: Variablen, Datentypen, Operatoren, Ausdrücke.

Aktivität: Der Lehrer sollte die act()-Methode der Klasse Enemy erklären, insbesondere wie lokale Variablen im Code verwendet werden, z. B. die Variable rotation. Der Lehrer sollte den Unterschied zwischen this.rotation und rotation beschreiben. Der Lehrer sollte das Verhalten der getOneIntersectingObject(_cls_)-Methode beschreiben. Eine Referenz des überlappenden Objekts wird dann in einer lokalen Variablen gespeichert (eine Klassenumwandlung per Casting ist hier erforderlich). Wenn kein Schnittpunktobjekt vorhanden ist, wird der NULL-Wert zurückgegeben. Basierend auf einer durchzuführenden booleschen Auswertung wird dann eine entsprechende Handlung ausgeführt (d.h. rotieren oder umdrehen). Die Ergebnisse werden von der Lehrkraft und den Mitschülern diskutiert.

Commit*:* 97dddc4beba40ac785c7413bb245ba849cd956d2.

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/97dddc4beba40ac785c7413bb245ba849cd956d2

6.3. Einführung in Klassen-Konstruktoren

6.3.1. Dokumentation des Unterrichtsszenarios

Tabelle 11. Einführung in Klassen-Konstruktoren

Titel	Einführung in Klassen-Konstruktoren
Lernziele	Am Ende dieser Unterrichtseinheit werden die Schüler in der Lage sein, Konstruktoren zu verstehen und anzuwenden. Dafür werden grundlegende theoretische Konzepte im Zusammenhang mit Konstruktoren vorgestellt, sowie Code erklärt und Aufgaben gelöst. Die untersuchten Konzepte werden im Kontext der Spieleentwicklung diskutiert. Daneben sollen auch Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren gefördert werden.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und die bisher im Kurs vermittelten Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten bereits mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Grundlegende Konzepte von Konstruktoren (10 Minuten) 2. Erklärung des Codes (20 Minuten) 3. Aufgabe: Umbenennen der Klasse MyWord in Arena (5 Minuten) 4. Aufgabe: Layout der Arena erstellen (30 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 65-minütigen Unterrichtseinheit werden Schülerinnen und Schüler der Sekundarstufe mit dem Greenfoot-Tool in Konzepte eingeführt, die mit Klassen-Konstruktoren im Rahmen der Spieleentwicklung zu tun haben. Es beginnt mit einer 10-minütigen Einführung durch den Lehrer, in der grundlegendes Wissen über Konstrukteuren vermittelt wird. Daran schließt sich ein 20-minütiger Teil an, in dem verschiedene Codebeispiele im Greenfoot-Umfeld praxisnah erklärt und diskutiert werden. Die nächste 5-minütige Einheit beinhaltet eine Aufgabe. Die zuvor definierte Klasse MyWorld soll dafür umbenannt werden. In diesem Zusammenhang sollte ein neuer Name definiert werden. Als Klassenname wird hier Arena verwendet. Dabei ist zu beachten, dass auch der Konstruktor der Klasse entsprechend

	umbenannt werden sollte. Darauf folgt ein 30-minütiger Abschnitt unter Anleitung des Lehrers, in der die Schüler Aufgaben lösen, die mit dem Layout der Arena zusammenhängen. Es ist Feedback von der Lehrkraft und von den Mitschülern zu berücksichtigen. Die Schülerinnen und Schüler werden die Arbeit an dem in den vorangegangenen Schritten initiierten Spielprojekt fortsetzen. Am Ende der Unterrichtseinheit sind die Konzepte von Klassen-Konstruktoren bekannt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Der Stand des Projekts eröffnet Möglichkeiten für Hausaufgaben. In diesem Zusammenhang können weitere Klassen, Ausdrücke und Werte eingeführt werden, um komplexeres Verhalten im Spiel zu erreichen (z.B. Teleports, Tunnel, etc.). Diese Konzepte können mit den Studierenden besprochen und die jeweilige Umsetzung als Hausaufgabe vergeben werden.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

6.3.2. Leitfaden zur Unterrichtsvorbereitung

1. Grundbegriffe von Konstruktoren

Gegenstand: Verständnis von Klassen-Konstruktoren.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Schlüsselworte: *super, new, this*.

Aktivität: Der Lehrer führt den Begriff *Konstruktor* im Kontext von Klassen- und Objekten ein. Konstruktoren werden verwendet, um eine konkrete Instanz (d. h. ein Objekt) einer Klasse zu initialisieren.

2. Erklärung des Codes

Gegenstand: Verständnis von Klassen-Konstruktoren anhand von Beispielen.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Methoden, Parameter, Schlüsselworte: *super, new*, *this*, Beispiele aus der Praxis für Konstruktoren.

Aktivität: Der Lehrer wird Konstruktoren im Kontext von Klassen und Objekten diskutieren: Konstruktoren werden verwendet, um konkrete Instanzen einer Klasse zu initialisieren. Darüber hinaus werden Konstruktoren automatisch bei der Erzeugung von Objekten aufgerufen und können entweder implizit oder explizit definiert werden. Es gibt Standardkonstruktoren (die

implizit definiert sind) sowie parametrisierte und nicht parametrisierte Konstruktoren (die explizit von einem Programmierer definiert werden). Die Unterschiede zwischen parametrisierten und nicht parametrisierten Konstruktoren sollten ebenfalls erörtert werden. Um dieses Konzept für die Schüler verständlicher zu machen, sollte der Lehrer Beispiele aus dem wirklichen Leben verwenden.

3. Aufgabe: Umbenennen der Klasse MyWord in Arena

Gegenstand: Umbenennung der MyWorld-Klasse.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte.

Aktivität: Die zuvor definierte Klasse MyWorld soll in diesem Schritt umbenannt werden. Der neue Name soll Arena sein. Zusätzlich muss dann auch der Konstruktor der Klasse von MyWorld() in Arena() umbenannt werden.

Commit: aaf73c9bfd9f76a2a1e504f5e78d2976f1cada12

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/aaf73c9bfd9f76a2a1e504f5e78d2976f1cada12

4. Aufgabe: Layout der Arena erstellen

Gegenstand: Verständnis von Konstruktoren und Bezug auf das Spielszenario.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Methoden, Parameter, Schlüsselwörter: *super, new, this*.

Aktivität: In dieser Aktivität soll ein benutzerdefinierter Anfangszustand mit verschiedenen platzierten Objekten für die Arena erstellt werden. Der Anfangszustand wird im Konstruktor der Arena-Klasse eingerichtet: Eine Instanz von Enemy, eine Instanz von Orb und mindestens eine Instanz von Direction werden hinzugefügt. Nach dem Deklarieren und Initialisieren der Objekte sollen Eigenschaften durch Aufrufen der entsprechenden Methoden zugewiesen werden. Schließlich müssen diese Objekte durch Aufrufen der addObject(Actor)-Methode in die Arena integriert werden.

Commit: 8b105ea2eaf697f08c321efe687ddd31e2d0a041

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/8b105ea2eaf697f08c321efe687ddd31e2d0a041

6.4. Einführung in Attribute in der Greenfoot-Umgebung

6.4.1. Dokumentation des Unterrichtsszenarios

Tabelle 12. Einführung in Attribute in der Greenfoot-Umgebung

Titel	Einführung in Attribute in der Greenfoot-Umgebung
Lernziele	Am Ende dieser Unterrichtseinheit werden die Schüler in der Lage sein, Attribute von Klassen zu verstehen. Das Konzept von Attributen wird eingeführt, verschiedene Code-Erklärungen vorgenommen und Aufgaben gestellt. Die Klassenattribute werden Kontext der Spieleentwicklung angwendet. Damit werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Aufgabe: Bewegungsbezogenes Problem und Lösung (30 Minuten) 2) Klassenattribute (10 Minuten) 3) Parameter von Konstruktoren (10 Minuten) 4) Aufgabe: Erweiterung der Klasse Enemy mit dem Attribut moveDelay zur Verögerung (20 Minuten) 5) Aufgabe: Bewegung mit Verzügerung (30 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	In dieser 100-minütigen Unterrichtseinheit werden Schülerinnen und Schüler der Sekundarstufe mit dem Greenfoot-Tool in Konzepte rund um Attribute von Klassen eingeführt. Der Unterricht beginnt mit einer 30-minütigen Aufgabe, die sich auf bewegungsbezogene Probleme und mögliche Lösungen bezieht. Basierend auf der vorherigen Aufgabe werden in der nächsten 10-minütigen Einheit Attribute eingeführt. Daran schließt sich ein 10-minütiger Teil an, in dem Parameter von Konstruktoren erläutert und diskutiert werden. Der nächste 20-minütige Abschnitt unter Anleitung des Lehrers bezieht sich auf eine Aufgabe. Es wird ein neues Attribut definiert, das sich auf die Bewegung in der Enemy-Klasse bezieht. Zusätzlich wird der parametrische Konstruktor definiert. Lehrkraft und Mitschüler geben Feedback. Im letzten 30-minütigen Teil wird unter Anleitung des Lehrers schließlich die Bewegung von Feinden implementiert. In diesem Zusammenhang wird die act()-

	Methode aktualisiert. Lehrkraft und Mitschüler geben Feedback. Die Schülerinnen und Schüler werden die Arbeit an dem in den vorangegangenen Unterrichtseinheiten programmierten Spielprojekt fortsetzen und dabei in die Konzepte der Klassen-Attribute eingeführt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Der Stand des Projekts eröffnet Möglichkeiten für Hausaufgaben. In diesem Zusammenhang können weitere Klassen, Ausdrücke und Werte eingeführt werden, um komplexeres Verhalten im Spiel zu erreichen (z.B. Teleports, Tunnel, etc.). Diese Konzepte können mit den Studierenden besprochen und die jeweilige Umsetzung als Hausaufgabe vergeben werden.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

6.4.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe: Bewegungsbezogenes Problem und Lösung

Gegenstand: Verständnis von bewegungsbezogenen Problemen und möglicher Lösungen im Kontext von Konstruktoren und Attributen.

Zu besprechende Konzepte: Konstruktoren, Attribute, Methoden.

Aktivität: Der Lehrer erklärt, dass ein Enemy-Objekt (Feind) sich immer zwei Zellen in jedem Schritt bewegt, was zu Problemen bei seiner Bewegung führt. Um dies zu beheben, kann die Geschwindigkeit des Feindes auf eine andere Weise modelliert werden. Die gegnerische Instanz bewegt sich jetzt immer um eine Zelle nach der anderen. Zusätzlich kann ein neues Attribut namens moveDelay definiert werden, das bewirkt, dass sich die Enemy-Instanz erst nach einer bestimmten Anzahl von act()- Methodenaufrufen bewegt.

2. Klassenattribute

Gegenstand: Verständnis des Konzepts von Attributen.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Attribute.

Aktivität: Der Lehrer führt in das Konzept der Attribute im Kontext von Klassen- und Objektkonzepten in der objektorientierten Programmierung (OOP) ein.

3. Parameter von Konstruktoren

Gegenstand: Verständnis der Parameter von Konstruktoren.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Attribute, Parameter, Schlüsselworte: *super, new, this*.

Aktivität: Der Lehrer führt in das Konzept von Konstruktorparametern im Kontext von Klassen und Objekten ein.

4. Aufgabe: Erweiterung der Klasse Enemy mit dem Attribut moveDelay

Gegenstand: Verständnis von Klassen-Attributen und Parametern von Konstruktors.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Attribute, Parameter, Schlüsselworte: *super, new, this*.

Aktivität: Ein neues Attribut mit dem Namen moveDelay vom Typ int wird der Enemy-Klasse hinzugefügt. Es wird auch ein parametrisierter Konstruktor definiert, um dieses Attribut zu initialisieren, wobei das Attribut auf den vom Parameter bereitgestellten Wert festgelegt wird. Der Code in der Arena-Klasse wird entsprechend angepasst.

Commit: 6092489ce57541e77ae4e2ee886b20853df9f8a4

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/6092489ce57541e77ae4e2ee886b20853df9f8a4

5. Aufgabe: Bewegung mit Verzögerung

Gegenstand: Verständnis von Klassen-Attributen und Parametern von Konstruktors.

Zu besprechende Konzepte: Konstruktoren, Klassen, Objekte, Attribute, Parameter, Schlüsselworte: *super, new, this*.

Aktivität: Die act()-Methode der Enemy-Klasse wird so angepasst, dass sich das Enemy-Objekt erst nach der angegebenen Anzahl von moveDelay-Aufrufen der Methode bewegt. Darüber hinaus wird ein neues Attribut mit dem Namen nextMoveCounter vom Typ int eingeführt und auf 0 initialisiert. Die act()-Methode wird so geändert, dass sie this.move(1) nur dann aufruft, wenn nextMoveCounter 0 erreicht. Nach Ausführen des Bewegungsschritts wird nextMoveCounter auf den Wert von moveDelay zurückgesetzt. Wenn sich die gegnerische Instanz noch nicht bewegen kann, weil nextMoveCounter noch nicht 0 erreicht hat, dann wird nextMoveCounter um 1 verringert.

Commit: bf26e6ed23911ccb712fae3e243cdedff3a89a7f

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/bf26e6ed23911ccb712fae3e243cdedff3a89a7f

6.5. Einführung in das Überladen von Konstruktoren

6.5.1. Dokumentation des Unterrichtsszenarios

Tabelle 13. Einführung in das Überladen von Konstruktoren

Titel	Einführung in die Überladung von Konstruktoren in der Greenfoot-Umgebung
Lernziele	Am Ende dieser Unterrichtseinheit werden die Teilnehmer in der Lage sein, Konstruktorüberladung zu verstehen. Im Unterricht werden grundlegende Konzepte zur Überladung von Konstruktoren erklärt, sowie verschiedene Codeerklärungen vorgenommen und Aufgaben gestellt. Die untersuchten Konzepte werden im Kontext der Spieleentwicklung diskutiert. Daneben werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Grundlegende Konzepte der Konstruktorüberladung (5 Minuten) 2. Aufgabe: Parametrisierbarer Konstruktor der Klasse Direction (25 Minuten) 3. Aufgabe: Überladung des Konstruktors der Klasse Direction (25 Minuten) 4. Wiederholung (20 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 75-minütigen Unterrichtseinheit werden Schülerinnen und Schüler der Sekundarstufe mit dem Greenfoot-Tool in das Überladen von Klassen-Konstruktoren eingeführt. Die Sitzung beginnt mit einer 5-minütigen Einführung in die grundlegenden Konzepte des Überladens von Konstruktoren. Der nächste 25-minütige Abschnitt unter Anleitung des Lehrers enthält eine Aufgabe, in der parametrierbare Konstruktor in der Klasse Direction definiert wird. In der nächsten 25-Minuten-Aufgabe wird der überladene parameterlose Konstruktor in der Klasse Direction definiert. Am Ende des Abschnitts wird Feedback vom Lehrer und von den Mitschülern gegeben.

	Schließlich wird in der letzten 20-minütigen Einheit unter Anleitung des Lehrers eine Wiederholung der zuvor besprochenen programmier-sprachlichen Techniken durchgeführt: Variable, Ausdrücke, Operatoren, Konstruktoren, Attribute und das Überladen von Konstruktoren.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Der Stand des Projekts eröffnet Möglichkeiten für Hausaufgaben. In diesem Zusammenhang können weitere Klassen, Ausdrücke und Werte eingeführt werden, um komplexeres Verhalten im Spiel zu erreichen (z.B. Teleports, Tunnel, etc.). Diese Konzepte können mit den Schülern besprochen und die jeweilige Umsetzung als Hausaufgabe vergeben werden.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

6.5.2. Leitfaden zur Unterrichtsvorbereitung

1. Grundlegende Konzepte der Konstruktorüberladung

Gegenstand: Verständnis der Konstruktorüberladung.

Zu besprechende Konzepte: Konstruktoren.

Aktivität: Besprechen der Konstruktorüberladung bezüglich Syntax und Wirkungsweise.

2. Aufgabe: Schreiben eines parametrisierbaren Konstruktors der Klasse Direction

Gegenstand: Schreiben des parametrisierbaren Konstruktors der Klasse Direction im Kontext der Spieleentwicklung.

Zu besprechende Konzepte: Konstruktoren, Parameter, Attribute, parametrisierbare Konstruktoren.

Aktivität: In diesem Abschnitt wird ein parametrisierter Konstruktor für die Direction-Klasse mit einem einzigen Parameter definiert, **rotation** vom Typ int. Innerhalb des Konstruktors sollte die erstellte Instanz basierend auf dem Wert dieses Parameters rotiert werden. Danach sollte der Code in der Arena-Klasse entsprechend aktualisiert werden.

Commit: 3c4b9ef57ab17bac2a0abc7fc5e76ea4b6e27e4b

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/3c4b9ef57ab17bac2a0abc7fc5e76ea4b6e27e4b

3. Aufgabe: Überladung des Konstruktors der Klasse Direction

Gegenstand: Definition des überladenen Konstruktors der Direction-Klasse im Kontext der Spieleentwicklung.

Zu besprechende Konzepte: Konstruktoren, Konstruktor-Überladung.

Aktivität: In diesem Teil wird ein überladener Konstruktor in der Direction-Klasse definiert. Es handelt sich dabei um einen nichtparametrisierten Konstruktor. Innerhalb seines Textkörpers wird der parametrisierte Konstruktor aufgerufen, wobei das Argument auf 0 festgelegt wird. Das entspricht einer Drehung von 0 Grad. Der Code in der Arena-Klasse sollte entsprechend aktualisiert werden, wobei nach Möglichkeit die nicht parametrisierte Version des Direction-Klassenkonstruktors verwendet werden sollte.

Commit: 1e67e67523c66acea4e93363c9a3173302f424c8

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/1e67e67523c66acea4e93363c9a3173302f424c8

4. Wiederholung

Gegenstand: Wiederholung der zuvor eingeführten Konzepte.

Zu besprechende Konzepte: Variablen, Ausdrücke, Operatoren, Konstruktoren, Attribute, Überladung von Konstruktoren.

Aktivität: In diesem Abschnitt werden die zuvor diskutierten Konzepte wiederholt und zusammengefasst.

7. Assoziation

Innerhalb der Themeneinheit Assoziation werden vier Unterrichtsszenarien dokumentiert.

7.1. Objekte in der Greenfoot-Umgebung: Erkunden von Methoden und Assoziation

7.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 14. Objekte in der Greenfoot-Umgebung: Erkunden von Methoden und Assoziation

Titel	Objekte in der Greenfoot-Umgebung: Erkunden von Methoden und Assoziation
Lernziele	Am Ende dieser Unterrichtseinheit sollten die Schülerinnen und Schüler ein solides Verständnis dafür haben, wie Objekte miteinander interagieren. Eine Instanz der Klasse Enemy (Feind) interagiert innerhalb der Greenfoot-Umgebung mit anderen Objekten, insbesondere der Klasse Orb (Krone, die angegriffen werden soll). Es werden Kenntnisse im Erstellen und Aufrufen von Methoden innerhalb von Java-Klassen entwickelt, insbesondere beim Implementieren und Testen der Methoden Arena.respawn(Enemy) und Orb.hit(Enemy). Darüber hinaus sollten die Schüler Klassenattribute verstehen und effektiv verwalten, einschließlich der Definition und Verwendung der Attribute Enemy.attack und Orb.hp. Die Teilnehmer sollten in der Lage sein, Daten innerhalb einer Klasse zu kapseln, was durch die Erstellung von Gettern wie getEnemy.attack() demonstriert wird, und die Bedeutung der Datenkapselung für sicheren und wartbaren Code verstehen. Sie sollten die Nachrichtenübergabe zwischen Objekten verstehen und implementieren können, um zu erreichen, dass Instanzen von Klassen effektiv kommunizieren und um Spielaktionen auszuführen. Darüber hinaus sollten die Schüler Methodenaufruftechniken anwenden, um interaktive Spielentwicklungsaufgaben zu lösen, indem sie die für den Methodenaufruf erforderliche Syntax und Parameter effektiv nutzen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse vorausgesetzt, einschließlich Iteration und Programmverzeigungen. Die Schüler sollten bereits mit Greenfoot vertraut sein.
Dauer des	1. Aufgabe Task 5.1 - Besprechen, was passieren soll, wenn der Feind die Krone erreicht (10 Minuten)

Szenarios	2. Aufgabe Task 5.2 - Besprechen, wie die Enemy-Objekte (Klasse Enemy, Feind) mit den relevanten Objekten interagieren sollen, indem Nachrichten verwendet werden, wenn die Instanz der Klasse Orb (Krone) berührt wird (15 Minuten) 3. Aufgabe Task 5.3 - Attribute Enemy.attack und Orb.hp (10 Minuten) 4. Methoden (15 Minuten) 5. Aufgabe Task 5.4 - Getter des Attributs Enemy.attack (5 Minuten) 6. Aufgabe Task 5.5 - Erstellen und Testen der Methode Arena.respawn(Enemy) (10 Minuten) 7. Aufgabe Task 5.6 – Erstellen und Testen der Methode Orb.hit(Enemy) (10 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	In diesem 75-minütigen Unterrichtsszenario tauchen Schülerinnen und Schüler der Sekundarstufe in die Objektinteraktionen, die Methoden-erstellung und den Umgang mit Attributen in der Greenfoot-Umgebung ein. Der Unterricht zielt darauf ab, das Verständnis der Schüler dafür zu verbessern, wie Objekte kommunizieren und interagieren. Das ist ein entscheidender Aspekt der objektorientierten Programmierung. Der Unterricht beginnt mit einer 10-minütigen Diskussion über die Dynamik zwischen feindlichen Objekten der Krone, in der untersucht wird, was passieren sollte, wenn ein Feind die Krone erreicht. Das ist eine Voraussetzung für das Verständnis von Objektinteraktionen in einem Spielkontext. Im Anschluss daran diskutieren die Schülerinnen und Schüler in einer 15-minütigen Aufgabe, wie eine Instanz der Klasse Feind mit relevanten Objekten mithilfe von Nachrichten interagieren sollte, wenn sie mit einer Orb-Objekt (Krone) kollidiert. In dieser Diskussion wird die Bedeutung der Objektkommunikation und der Nachrichtenweitergabe hervorgehoben. Als nächstes wird sich in einer 10-minütigen Aufgabe auf die Attribute Enemy.attack und Orb.hp konzentriert. Die Schüler werden diese Attribute definieren und verstehen, wie diese für die Verwaltung der Spielmechanik entscheidend sind. In den folgenden 15 Minuten lernen die Schülerinnen und Schüler, wie sie Methoden erstellen und in ihren Klassen implementieren können. In diesem

	Abschnitt wird ihr Verständnis der Methodenerstellung und des Methodenaufrufs gefestigt. In einer 5-minütigen Aufgabe müssen die Schüler einen Getter für das Attribut Enemy.attack erstellen, um ihr Wissen über Kapselung und Datenabruf zu vertiefen. Die nächsten 10 Minuten werden dem Erstellen und Testen der Methode Arena.respawn(Enemy) gewidmet. Die Teilnehmer werden diese Methode implementieren, um das Wiedereinsetzen feindlicher Objekte zu handhaben und so ihr Verständnis der Methodenfunktionalität und des Testens sicherzustellen. Danach wird eine weitere 10-minütige Aufgabe das Erstellen und Testen der Methode Orb.hit(Enemy) beinhalten, die die Interaktion steuert, wenn ein Feind die Krone erreicht. Während des Unterrichts arbeiten die Schüler einzeln oder in kleinen Gruppen, um die Zusammenarbeit und das Peer-Learning zu fördern. Durch die aktive Teilnahme an Diskussionen, Programmieraufgaben und das Testen von Methoden entwickeln die Schüler Fähigkeiten zum algorithmischen Denken und zur Lösung von Problemen. Am Ende der Unterrichtseinheit haben die Schüler ein umfassendes Verständnis der Objektinteraktionen, der Methodenerstellung und des Attributmanagements in Greenfoot erhalten. Diese grundlegenden Fähigkeiten werden sie für weitere Entwicklungsprojekte rüsten und ihre allgemeinen Programmierkenntnisse verbessern.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

7.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 5.1 - Besprechen, was passieren soll, wenn der Feind die Krone erreicht

Gegenstand: Es wird eine Diskussion vom Lehrer moderiert. Dabei geht es um das zu erwartende Verhalten, wenn ein Feind die Krone erreicht. Die Schüler sollen über die Spieldynamik und die Ergebnisse nachdenken, zum Beispiel über die Krone, die Schaden nimmt, oder das Ende des Spiels.

Zu besprechende Konzepte: Beschädigung der Krone, Entfernung von Gegnern, Auslösen von Spielereignissen, z. B. Verringerung der Spielstärke, Abspielen von Soundeffekten, Beenden des Spiels.

Aktivität: Die Unterrichtseinheit beginnt mit einer Wiederholung der zuvor behandelten Konzepte, um sicherzustellen, dass die Schüler eine solide Grundlage für das neue Material haben. Der Lehrer diskutiert mit den Schülern, um die Konzepte der Objektinteraktion, Botschaften und Methoden im Kontext der Greenfoot-Umgebung zu klären.

Die Schüler arbeiten in Gruppen für ein Brainstorming zusammen, um das Verhalten festzulegen, wenn ein Feind die Krone in ihrem Spiel erreicht. Sie erörtern mögliches Verhalten, wie das Reduzieren der Spielstärke (Health Points, HP) der Krone bei Feindkontakt und diskutieren Szenarien, in denen die HP der Krone auf Null fallen könnte, was zum Ende des Spiels führen würde. Wenn die HP der Krone über Null bleibt, ist es möglich, den Feind an einem anderen Arenaort neu erscheinen zu lassen. Sie erwägen auch, zusätzliche Spielereignisse zu integrieren, die durch diese Interaktion ausgelöst werden, wie z. B. Soundeffekte oder Bildschirmmeldungen. Während der gesamten Gruppenarbeit leitet der Lehrer die Diskussion und fordert die Schüler auf, ihre vorgeschlagenen Algorithmen auf ein bestimmtes Szenario auszurichten: Sicherzustellen, dass die HP der Krone abnimmt, wenn sie von einem Feind erreicht wird, wie zuvor beschrieben. Diese Anleitung hilft den Schülern, ihre Ideen praktisch anzuwenden und ihr Verständnis der Spieldynamik und der Wechselwirkungen innerhalb des Rahmens ihres Spiels zu stärken.

2. Aufgabe Task 5.2 - Besprechen, wie die Enemy-Objekte (Klasse Enemy, Feind) mit den relevanten Objekten interagieren sollen, indem Nachrichten verwendet werden, wenn die Instanz der Klasse Orb (Krone) berührt wird.

Gegenstand: Der Lehrer erarbeitet mit den Schülern das Verständnis, wie eine Instanz der Enemy-Klasse mit anderen Objekten interagieren sollte, insbesondere wenn sie auf eine Instanz der Orb-Klasse (Krone) auf dem Spielfeld trifft.

Zu besprechende Konzepte: Methodenaufruf, Parameterübergabe, Objektreferenzen.

Aktivität: Die Schülerinnen und Schüler arbeiten in Paaren zusammen, um einen Vorschlag zu erarbeiten, wie eine Instanz der Enemy-Klasse mit einer Instanz der Orb-Klasse interagieren sollte, indem sie Nachrichten in ihrem Spielszenario verwenden. Sie analysieren und skizzieren die Abfolge von Nachrichten und Aktionen, die sich ergeben sollen, wenn ein Feind auf die Krone trifft. Diese Übung zielt darauf ab, ihr Verständnis von Methodenaufrufen, Parameterübergabe und Objektreferenzen im Kontext der Spieleentwicklung zu vertiefen. Während die Schüler ihre Ideen diskutieren und verfeinern, leitet der Lehrer sie an, dass die Abfolge der Interaktionen zwischen den Objekten dem Algorithmus folgt, der auf kooperierende Objekte verteilt ist, wie in den Zielen

der Unterrichtseinheit angegeben. Um diesen Prozess zu unterstützen, kann der Lehrer ein UML-Sequenzdiagramm einführen und verwenden, um die Interaktionen zwischen den Klassen Enemy, Orb, Arena und Greenfoot visuell zu beschreiben. Dieses visuelle Tool hilft den Schülern, den Fluss von Nachrichten und Methodenaufrufen besser zu verstehen und stärkt ihr Verständnis für objektorientierte Programmierkonzepte und deren Anwendung in der Java-Programmierung innerhalb der Greenfoot-Umgebung.

3. Aufgabe Task 5.3 – Attribute Enemy.attack und Orb.hp

Gegenstand: Der Lehrer stellt die Attribute Enemy.attack und Orb.hp vor und erklärt ihre Bedeutung für das Ergebnis der Interaktionen.

Zu besprechende Konzepte: Klassenattribute, Kapselung.

Aktivität: Der Lehrer führt in das Konzept der Klassenattribute und der Kapselung ein und erklärt, dass Attribute wie Enemy.attack und Orb.hp die Eigenschaften von Objekten repräsentieren. Die Schülerinnen und Schüler lernen, diese Attribute in Klassen zu definieren und im Programm zu verwenden, beisielsweise um die Angriffskraft des Feindes und die Spielstärke (hp) der Krone zu modellieren. Sie beginnen mit dem Hinzufügen eines neuen ganzzahligen Attributs attack zur Enemy-Klasse, einschließlich eines Parameters im Konstruktor zum Initialisieren dieses Attributs. Auf ähnliche Weise fügen sie der Orb-Klasse ein ganzzahliges Attribut namens hp hinzu, zusammen mit einem parametrisierbaren Konstruktor, um diesen Wert bei der Objekterstellung festzulegen. Der Lehrer leitet die Schüler an, den Code in der Arena-Klasse anzupassen, um diese neuen Attribute zu berücksichtigen. Diese praktische Erfahrung hilft den Teilnehmern, die Rolle von Klassenattributen bei Objektinteraktionen und die Bedeutung der Kapselung für die Aufrechterhaltung der Codeintegrität und -sicherheit zu verstehen.

Commit: 4ca1e9f25685990d2bdfe5b610c28422e0944f95

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/4ca1e9f25685990d2bdfe5b610c28422e0944f95

4. Methoden

Gegenstand: Die Lehrkraft gibt einen Überblick über die Methoden und erklärt, wie sie verwendet werden, um Aktionen und Verhaltensweisen der Objekte zu kapseln.

Zu besprechende Konzepte: Methodendefinition, Methodenaufruf, Parameter, Rückgabewerte.

Aktivität: Der Lehrer beginnt damit, Methoden als gekapselte Aktionen oder Verhaltensweisen innerhalb einer Klasse zu erklären. Anhand von praktischen Beispielen demonstriert der Lehrer die Syntax und Struktur von Methodendefinitionen und veranschaulicht, wie Methoden von Objekten aufgerufen werden. Die Schüler lernen verschiedene Arten von Methoden kennen, einschließlich solcher, die Aktionen ausführen (void-Methoden) und solcher, die Werte zurückgeben (return-type-Methoden). Der Lehrer erklärt, wie Parameter an Methoden übergeben werden und hebt die Wichtigkeit von Parametertypen und deren Reihenfolge hervor. Anhand von geführten Codierungsaufgaben üben die Teilnehmer das Definieren von Methoden mit verschiedenen Parameter- und Rückgabetypen und das Aufrufen dieser Methoden für Objekte. Sie untersuchen

Szenarien, in denen Methoden Aktionen ausführen, Objektzustände ändern oder bestimmte Werte zurückgeben, um ihr Verständnis der Methodenfunktionalität innerhalb einer Klasse zu festigen.

5. Aufgabe Task 5.4 – Getter des Attributs Enemy.attack

Gegenstand: Der Lehrer erklärt das Konzept der Getter-Methoden und deren Benutzung beim Zugriff auf Attributwerte.

Zu besprechende Konzepte: Getter-Methoden, Kapselung.

Aktivität: Der Lehrer erklärt den Zweck von Getter-Methoden und betont, wie sie einen kontrollierten Zugriff auf Attributwerte ermöglichen, während die Kapselung beibehalten wird. Die Schülerinnen und Schüler lernen, dass es wichtig ist, Getter zum Abrufen privater Attributwerte zu verwenden, wodurch das Konzept des Zugriffsschutzes innerhalb einer Klasse gestärkt wird. Der Lehrer führt die Schüler dann durch den Prozess der Erstellung einer Getter-Methode für das attack-Attribut in der Enemy-Klasse. Mit einem praxisorientierten Ansatz implementieren die Studierenden die Getter-Methode und stellen sicher, dass sie den Wert des attack-Attributs zurückgibt. Der Lehrer demonstriert die korrekte Syntax und Struktur zum Definieren eines Getters und wie er innerhalb des Codes verwendet wird, um auf den Attributwert zuzugreifen. Danach sollten die Studierenden in der Lage sein, Getter-Methoden zu erstellen und zu nutzen, um kontrolliert auf Attributwerte zuzugreifen.

Commit*:* 72b7456ea4cc11416c57d72c89b6a7f7e9266e3e

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/72b7456ea4cc11416c57d72c89b6a7f7e9266e3e

6. Aufgabe Task 5.5 – Erstellen und Testen der Methode Arena.respawn(Enemy)

Gegenstand: Der Lehrer führt die Schüler durch die Erstellung und das Testen der Methode Arena.respawn(Enemy), die das wiederholte Erzeugen von Gegnern im Spiel übernimmt.

Zu besprechende Konzepte: Methodenimplementierung, Testen, Spielmechanik.

Aktivität: Der Lehrer beginnt mit einer Einführung die Methodenimplementierung und deren Bedeutung bei der Definition spezifischer Verhaltensweisen innerhalb einer Klasse. Mit dem Schwerpunkt auf praktische Anwendung werden die Schüler in der Arena-Klasse angeleitet, die Respawn-Methode zu erstellen. Diese Methode, die keinen Wert zurückgibt, akzeptiert einen einzelnen Parameter vom Typ Enemy. Die Schüler werden beauftragt, die Position und Drehung des Gegners innerhalb dieser Methode so einzustellen, dass sie den ursprünglich im Konstruktor festgelegten Werten entsprechen. Der Lehrer demonstriert die korrekte Syntax und Struktur für die Definition dieser Methode und beschreibt damit noch einmal das Prinzip der Methodenimplementierung und Parameterübergabe.

Als nächstes testen die Schülerinnen und Schüler ihre Methode, um zu prüfen, dass sie korrekt funktioniert. Sie erstellen eine Instanz der Arena und einen Feind, starten die Anwendung aber

nicht sofort. Stattdessen ziehen sie die Gegner-Instanz an einen neuen Ort und greifen dann auf das Kontextmenü der Arena-Instanz zu, um die Respawn-Methode aufzurufen. Der Lehrer erklärt, wie man erreicht, dass die Anwendung pausiert und das Parameterfeld aktiv ist, und leitet die Schüler an, mit der rechten Maustaste auf die Enemy-Instanz zu klicken, um das Parameterfeld korrekt auszufüllen. Die Schüler beobachten den im Fenster erstellten Ausdruck und klicken dann auf die Schaltfläche OK, um die Auswirkungen ihrer Respawn-Methode zu sehen.

Durch diese Aktivität sammeln die Schülerinnen und Schüler praktische Erfahrungen im Schreiben und Testen von Methoden und verstehen, wie Spielobjekte programmgesteuert manipuliert werden können. Der Lehrer kümmert sich darum, dass die Schüler jeden Schritt verstehen, und bietet bei Bedarf Hilfestellung und Erklärung an, um das Verständnis für die Implementierung der Methode und die Spielmechanik zu stärken.

Commit*:* 43a221876b8acb4fd507175ec4c8f520121d1ab1

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/43a221876b8acb4fd507175ec4c8f520121d1ab1

7. Aufgabe Task 5.6 - Erstellen und Testen der Methode Orb.hit(Enemy)

Gegenstand: Der Lehrer leitet die Schüler an, die Methode Orb.hit(Enemy) zu erstellen und zu testen. Diese Methode definiert, was passiert, wenn ein Feind die Krone erreicht.

Zu besprechende Konzepte: Methodeninteraktion, Aktualisieren des Objektzustands.

Aktivität: Der Lehrer beginnt damit, den Zweck der Orb.hit(Enemy)-Methode zu erklären und hebt hervor, wie sie die Interaktion zwischen der Krone und einem Feind herstellt. Die Schüler werden dann angeleitet, diese Methode zur Orb-Klasse hinzuzufügen. Diese Methode, die keinen Wert zurückgibt, nimmt einen einzelnen Parameter vom Typ Enemy entgegen.

Um die Methode zu testen, folgen die Schüler einem Schritt-für-Schritt-Vorgehen, das dem Testen der Respawn-Methode ähnelt. Sie erstellen eine Instanz der Klasse Orb und eine Instanz der Klasse Enemy. Ohne die Anwendung zu starten, rufen sie das Kontextmenü der Orb-Instanz auf und wählen die hit()-methode aus. Der Lehrer unterstützt die Schüler, wenn sie das Parameterfeld ausfüllen. Das geschieht wenn sie der rechten Maustaste auf die Enemy-Instanz klicken, während die Anwendung pausiert ist. Mit dieser Aktion wird der Ausdruck für den Methodenaufruf vorbereitet, den die Teilnehmer dann ausführen, indem sie auf die Schaltfläche OK klicken.

Der Lehrer betont, wie wichtig es ist, den im Fenster erstellten Ausdruck zu beobachten, um den Methodenaufruf zu überprüfen. Diese Übung hilft den Schülern, die Methodeninteraktion und die Aktualisierung von Objektzuständen innerhalb eines Spielkontexts zu verstehen. Durch diese praktische Aktivität sammeln die Schüler praktische Erfahrungen in der Implementierung und Erprobung von Methoden und vertiefen so ihr Verständnis für Methodeninteraktion und Spielverhalten. Der Lehrer unterstützt, erklärt bei Bedarf und bewirkt, damit die Schüler die Aufgabe erfolgreich abschließen und verstehen.

Commit*:* fe03d520260f172066be35055a901487bf7c2ff7

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/fe03d520260f172066be35055a901487bf7c2ff7

7.2. Objekte in der Greenfoot-Umgebung: Kennenlernen von Assoziation und fortgeschrittene Methodenaufrufe

7.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 15. Objekte in der Greenfoot-Umgebung: Assoziation und fortgeschrittene Methodenaufrufe

Titel	Objekte in der Greenfoot-Umgebung: Kennenlernen von Assoziation und fortgeschrittene Methodenaufrufe
Lernziele	Im Ergebnis diese Unterrichtsszenarios sollten die Schülerinnen und Schüler ein tiefes Verständnis dafür entwickeln, wie Objekte von verschiedenen Klassen Assoziationen bilden und in der Greenfoot-Umgebung interagieren. Die Schülerinnen und Schüler sollten Kenntnisse im Aufrufen der Orb.hit(Enemy)-Methode aus der Enemy-Klasse nachweisen und ihre Fähigkeit unter Beweis stellen, fortgeschrittene Methodenaufrufe zu initiieren, um die Interaktion zwischen Objekten herzustellen. Sie sollten dann auch in der Lage sein, die Funktionalität wichtiger Greenfoot-Methoden wie Greenfoot.stop() und World.getWorldOfType(_cls_) zu erklären, sowie deren Rolle bei der Steuerung der Spielausführung und der Verwaltung von Objektinstanzen zu verstehen. Die Schüler sollten die Orb.hit(Enemy)-Methode erfolgreich in ihren Projekten implementieren und Objektinteraktionen und Methodenfunktionalitäten integrieren, um interaktive und dynamische Spielmechaniken zu erstellen. Die Schüler sollten grundlegende Prinzipien der objektorientierten Programmierung, einschließlich Kapselung und Methodenaufruf, anwenden, um anspruchsvolle Spielinteraktionen und Funktionalitäten zu entwickeln. Sie sollen praktische Einblicke in verschiedene Aspekte der Spielentwicklung erhalten, einschließlich des Erzeugen von Gegner-Objekten, der Verwaltung des Spielzustands und der Schaffung ansprechender Spielerlebnisse durch Objektinteraktionen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse vorausgesetzt, einschließlich Iteration und Programmverzeigungen. Die Schüler sollten bereits mit Greenfoot vertraut sein.
Dauer des	1) Erklärung von Assoziation zwischen Objekten (10 Minuten) 2) Aufgabe Task 5.7 – Aufruf der Methode Orb.hit(Enemy) aus der Klasse

Szenarios	Enemy (15 Minuten) 3) Erklärung der Funktionalität der Methoden Greenfoot.stop() und World.getWorldOfType(_cls_) (15 Minuten) 4) Aufgabe Task 5.8 – Programmierung der Methode Orb.hit(Enemy) (30 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	In diesem 70-minütigen Unterrichtsszenario vertiefen sich Schülerinnen und Schüler der Sekundarstufe in die Details von Assoziationen zwischen Objekten und fortgeschrittenen Methodenaufrufen in der Greenfoot-Umgebung. Das Ziel des Unterrichts ist, das Verständnis für objektorientierte Programmierkonzepte zu vertiefen und die Fähigkeit zu verbessern, komplexe Interaktionen zu implementieren. Der Unterricht beginnt mit einer 10-minütigen Diskussion über Assoziationen zwischen Klassen, die sich darauf konzentriert, wie Objekte innerhalb eines Softwaresystems interagieren und zusammenarbeiten können. Dieses grundlegende Verständnis schafft die Voraussetzungen für die Implementierung komplexerer Wechselwirkungen. Im Anschluss daran werden die Schülerinnen und Schüler in 15 Minuten eine Aufgabe lösen, die darauf abzielt, die Methode Orb.hit (Enemy) aus der Enemy-Klasse aufzurufen. Diese Aufgabe konzentriert sich auf die praktische Anwendung des Methodenaufrufs und die Interaktion zwischen Objekten. Der nächste Abschnitt beinhaltet eine detaillierte Erklärung des Codes für die Methoden Greenfoot.stop() und World.getWorldOfType(_cls_), die 15 Minuten dauern soll. Die Schüler erhalten Einblicke, wie diese Methoden innerhalb der Greenfoot-Umgebung funktionieren und eine Steuerung der Spielelemente und des Weltmanagements ermöglichen. Der Kern dieser Unterichtseinheit ist eine 30-minütige Aufgabe, bei der die Schülerinnen und Schüler die Methode Orb.hit (Enemy) anwenden. Diese Aufgabe fordert die Schüler heraus, ihr Können der Methoden-implementierung, der Parameterübergabe und der Objektinteraktionen anzuwenden, um eine funktionierende Spielmechanik innerhalb ihrer Projekte zu erstellen. Am Ende erhalten die Schülerinnen und Schüler ein tieferes Verständnis der Assoziationen zwischen Objekten, Kenntnisse in fortgeschrittenen Methodenaufrufen wie Orb.hit(Enemy) aus der Enemy-Klasse und Einblicke in die Implementierung wichtiger Greenfoot-Methoden. Diese Fähigkeiten werden sie

	in die Lage versetzen, interaktivere und dynamischere Spiele zu entwickeln und das volle Potenzial objektorientierter Programmierprinzipien in der Spieleentwicklung auszuschöpfen.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

7.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Erklärung von Assoziation zwischen Objekten

Gegenstand: Es wird das Konzept der Assoziation zwischen Objekten erklärt, insbesondere wie Objekte verschiedener Klassen miteinander interagieren können.

Zu besprechende Konzepte: Assoziationen, Objektinteraktionen, Klassenbeziehungen.

Aktivität: Der Unterricht beginnt mit einem kurzen Überblick über Assoziationen zwischen Klassen in der objektorientierten Programmierung. Die Lehrkraft verwickelt die Schülerinnen und Schüler in eine Diskussion, um zu klären, wie Objekte durch Assoziationen miteinander interagieren, und veranschaulicht diese Konzepte anhand von praktischen Beispielen aus der Greenfoot-Umgebung. Die Schüler entwickeln ein Verständnis dafür, dass Assoziationen die Zusammenarbeit von Objekten definieren, z. B. wie ein Feind eine Krone in einem Spielszenario beeinflusst.

In einer ausführlichen Diskussion geht der Lehrer auf die verschiedenen Arten von Assoziationen von Klassen ein: eins-zu-eins (1:1), eins-zu-viele (1:n) und viele-zu-viele (n:m). Zum Beispiel könnte eine Eins-zu-Eins-Assoziation veranschaulichen, wie ein Spieler mit seinem Charakter-Avatar verbunden ist, eine Eins-zu-Viele-Assoziation könnte zeigen, wie eine Welt mehrere Actor-Objekte enthält, und eine m:n-Assoziation könnte darstellen, wie verschiedene Feinde mit mehreren Kronen in einem Spiellevel interagieren.

Anhand von UML-Klassendiagrammen, die speziell auf die in Greenfoot üblichen Klassen zugeschnitten sind, veranschaulicht der Lehrer diese Beziehungen visuell und hilft den Schülern zu verstehen, wie Assoziationen in ihren Spielprojekten umgesetzt werden. Das Diagramm könnte z. B. darstellen, wie ein Gegnerobjekt (Enemy) mit mehreren Instanzen der Krone-Klasse (Orb)

verknüpft ist, was auf eine 1:n-Beziehung hinweist, bei der jeder Feind mehrere Kronen (Orb-Objekte) beeinflusst.

Die Schülerinnen und Schüler beteiligen sich aktiv an der Identifizierung und Kartierung dieser Assoziationen in ihren Spielprojekten. Sie untersuchen, wie Objekte auf der Grundlage dieser Beziehungen interagieren und diskutieren die Implikationen für Spielmechanik und Logik. Der Lehrer stellt konkrete Beispiele aus dem Kontext der Spielentwicklung vor und veranschaulicht, wie Enemy-Instanzen mit Orb-Instanzen interagieren und wie diese Interaktionen durch Assoziationen ausgedrückt werden.

Als Ergebnis erhalten die Schüler ein solides Verständnis für die Rolle, die Assoziation bei der Gestaltung und Implementierung interaktiver Systeme spielen. Sie können verschiedene Arten von Assoziationen identifizieren und dieses Wissen anwenden, um komplexe Wechselwirkungen zwischen Greenfoot-Objekten effektiv zu modellieren und umzusetzen. Dieses praktische Verständnis stärkt ihre Fähigkeit, zusammenhängende und interaktive Spielszenarien nach objektorientierten Prinzipien zu entwerfen.

2. Aufruf Task 5.7 – Aufruf der Methode Method Orb.hit(Enemy) aus der Klasse Enemy

Gegenstand: Der Lehrer leitet die Schüler für das Aufrufen der Orb.hit(Enemy)-Methode aus der Enemy-Klasse an.

Zu besprechende Konzepte: Methodenaufruf, Objektreferenzen.

Aktivität: Die Schüler beteiligen sich an der praktischen Arbeit, indem sie die act()-Methode der Enemy-Klasse ändern. Sie entfernen vorhandenen Code, der für unnötige Verhaltensweisen verantwortlich ist, wie z. B. das Rotieren beim Auftreffen auf die Krone oder das Abprallen von den Rändern der Welt. Stattdessen implementieren sie die Funktionalität zum Aufrufen der Orb.hit(Enemy)-Methode, wenn eine Instanz von Enemy mit einer Instanz von Orb kollidiert.

Anhand von praktischen Beispielen und Demonstrationen veranschaulicht die Lehrkraft, wie der Methodenaufruf richtig eingerichtet wird. Die Schüler lernen, Objektreferenzen (this) zu verwenden, um die hit()-Methode auf der Orb-Instanz bei einer Kollision mit einem Feind auszulösen. Sie untersuchen den Kontrollfluss in Greenfoot und verstehen, wie der Methodenaufruf den Ausführungspfad innerhalb ihres Spielszenarios steuert.

Während der gesamten Zeit gibt der Lehrer Anleitungen zum Debuggen und Testen der Implementierung, um zu erreichen, dass der Aufruf von Orb.hit(Enemy) wie beabsichtigt funktioniert. Die Schülerinnen und Schüler beobachten das Verhalten in der Greenfoot-Umgebung und sehen, dass der Methodenaufruf die erwarteten Interaktionen zwischen Feind- und Krone-Objekten auslöst (Enemy und Orb).

Als Ergebnis der Unterrichtseinheit haben die Schülerinnen und Schüler Kenntnisse im Methodenaufruf und in der Objektinteraktion innerhalb der Greenfoot-Umgebung erworben. Sie verstehen jetzt, wie sie Objektreferenzen verwenden können, um Methoden über verschiedene Klassen hinweg aufzurufen, und vertiefen ihr Verständnis der objektorientierten Programmierung im Kontext der Spieleentwicklung.

Commit*:* 63f9c96717d9d2587b60095e3b249b0158c8587b

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/63f9c96717d9d2587b60095e3b249b0158c8587b

3. Erklärung der Funktionalität der Methoden Greenfoot.stop() und World.getWorldOfType(cls)

Gegenstand: Der Lehrer erklärt die Funktionsweise der Methode Greenfoot.stop() und der Methode World.getWorldOfType(_cls_).

Zu besprechende Konzepte: Methoden zur Steuerung des Spiels, Weltmanagement.

Aktivität: In diesem Abschnitt vertiefen sich die Schülerinnen und Schüler in das Verständnis von zwei wichtigen Methoden innerhalb des Greenfoot-Frameworks: Greenfoot.stop() und World.getWorldOfType(_cls_). Der Lehrer erklärt den Zweck und die Anwendung jeder Methode und ihre Rolle bei der Spielsteuerung und dem Weltmanagement.

Die Methode Greenfoot.stop() ist für die Steuerung des Ausführungsablaufs eines Greenfoot-Szenarios unerlässlich. Wenn sie aufgerufen wird, stoppt sie die Simulation und friert alle Akteure und Interaktionen innerhalb der Welt ein. Diese Methode ist besonders nützlich, um die Funktion zum Anhalten des Spiels zu implementieren oder bestimmte Ereignisse auszulösen, die ein vorübergehendes Anhalten des Spielfortschritts erfordern.

Andererseits dient die World.getWorldOfType(_cls_)-Methode einem weiteren Zweck, der mit der Verwaltung der Welt (World) zusammenhängt. Diese Methode ermöglicht es Entwicklern, Instanzen von Welten abzurufen, die einen bestimmten Klassentyp _cls_ aufweisen. Er durchläuft alle aktiven Welten in der Greenfoot-Umgebung und gibt Instanzen der Weltklasse zurück, die dem angegebenen Typ entsprechen. Diese Funktion ist von Vorteil, wenn Entwickler dynamisch mit Welten interagieren oder diese basierend auf ihren Klassenattributen bearbeiten müssen.

Die Schüler vertiefen anhand von praktischen Demonstrationen und Beispielen ihr Verständnis für diese Methoden. Der Lehrer zeigt, wie Greenfoot.stop() in Spielszenarien integriert werden kann, um Pausenfunktionen zu erstellen oder um ausgewählte Ereignisse im Spiel auszulösen. Die Schülerinnen und Schüler beobachten, wie sich das Anhalten des Spiels auf das Verhalten und die Interaktionen der Actor-Objekte in der Greenfoot-Umgebung auswirkt.

Auf ähnliche Weise erkunden die Schülerinnen und Schüler die Methode World.getWorldOfType(_cls_) durch praktische Übungen. Sie lernen, wie sie diese Methode verwenden, um Instanzen bestimmter Welttypen dynamisch abzurufen. Der Lehrer demonstriert Szenarien, in denen das Abrufen von Welten eines bestimmten Klassentyps erforderlich ist, um fortgeschrittene Spielmechaniken zu implementieren oder mehrere Spielumgebungen gleichzeitig in Greenfoot zu verwalten.

Während des gesamten Abschnitts regt der Lehrer zu Diskussionen an und stellt praktische Programmierbeispiele zur Verfügung, um die Anwendungen dieser Methoden in realen Spielentwicklungsszenarien zu veranschaulichen. Die Studierenden beteiligen sich aktiv am

Experimentieren mit den Methoden im Rahmen ihrer eigenen Greenfoot-Projekte und vertiefen ihr Verständnis durch direkte Anwendung und Beobachtung der Ergebnisse.

Als Ergebnis der Unterrichtseinheit haben die Schüler Kenntnisse in der Verwendung von Greenfoot.stop() für die Spielsteuerung und für World.getWorldOfType(_cls_) für ein effizientes Weltmanagement in der Greenfoot-Umgebung erworben. Damit verbessern sie ihre Fähigkeit, komplexes Spielverhalten zu implementieren und Spielzustände mit diesen wesentlichen Methoden effektiv zu verwalten.

4. Aufgabe Task 5.8 – Programmierung der Methode Orb.hit(Enemy)

Gegenstand: Der Lehrer führt die Schüler durch die Implementierung der Orb.hit(Enemy)-Methode.

Zu besprechende Konzepte: Methodenimplementierung, Aktualisierung des Objektzustands, Spielmechanik.

Aktivität: Die Schüler beginnen die Implementierung der Orb.hit(Enemy)-Methode. Mit dieser Methode wird ein entscheidender Schritt bei der Programmierung der Interaktion zwischen einem Feind und der Krone ausgeführt.

Die Orb.hit(Enemy)-Methode spielt eine entscheidende Rolle für das Verhalten, wenn ein Feind im Spiel mit der Krone kollidiert. Hier wird beschrieben, wie Schülerinnen und Schüler sich dieser Methode nähern und diese umsetzen können. Zunächst soll die Spielstärke des Krone-Objekts verringert werden. Diese Aktion simuliert den Schaden, den die Krone beim Aufprall eines Feindes erleidet. Danach wird überprüft, ob die Spielstärke der Krone auf null gefallen ist. Wenn diese einen Wert von Null oder weniger erreicht, dann sollte das Spiel beendet werden. Dies wird erreicht, indem Greenfoot.stop() aufgerufen wird. Das ist das Game Over-Szenario. Ein anderes Szenario liegt vor, wenn die Spielstärke der Krone nach der Kollision mit dem Feind immer noch über Null liegt. In diesem Fall sollten die Schüler Logik einbauen, den Feind wieder in die Arena hineinzusetzen, um ein kontinuierliches Weiterspielen zu ermöglichen.

Während der Aktivität unterstützt der Lehrer die Implementierung von Orb.hit (Enemy), indem er Anleitungen zur Methodenstruktur, zur Parameterverwendung (Enemy) und zur Aktualisierung des Zustands des Orb-Objekts gibt. Die Schüler diskutieren und entscheiden gemeinsam über die spezifischen Spielmechaniken, die sie implementieren möchten, z. B. wie viel Schaden jeder Gegnertyp an der Krone anrichtet und was passiert, wenn die Spielstärke der Krone aufgebraucht ist.

Der Lehrer begleitet die Schüler dabei, ihre Implementierungen gründlich zu testen und sicherzustellen, dass sich die Methode in verschiedenen Spielszenarien wie erwartet verhält. Als Ergebnis sammeln die Schüler praktische Erfahrungen in der Implementierung von Methoden, um Spielinteraktionen effektiv zu verwalten, und stärken ihr Verständnis der Methodenimplementierung und der Spielmechanik innerhalb der Greenfoot-Umgebung.

Commit*:* 84bcd7c128faaa9313b507f7438f826ae2f47d2c

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/84bcd7c128faaa9313b507f7438f826ae2f47d2c

7.3. Objekte in der Greenfoot-Umgebung: Türme, Geschosse und strategische Interaktionen

7.3.1. Dokumentation des Unterrichtsszenarios

Tabelle 16. Objekte in der Greenfoot-Umgebung: Türme, Geschosse und strategische Interaktionen

Titel	Objekte in der Greenfoot-Umgebung: Türme, Geschosse und strategische Interkationen
Lernziele	Als Ergebnis dieser Unterrichtseinheit entwickeln die Schüler Kenntnisse in der Erstellung der Bullet- und Tower-Klassen (Kugel und Verteidigungsturm) und schaffen so eine Grundlage für die strategische Spielentwicklung. Sie verstehen und implementieren realistische Kugelbewegungen und bestimmen Aktionen, wenn Kugelinstanzen auf feindliche Instanzen oder den Rand der Arena treffen. Die Teilnehmer entwerfen effektive Schießmechaniken für Turmobjekte und nutzen die Nachrichtenweitergabe zwischen Turmobjekten und anderen Spielelementen, um die Spieldynamik zu verbessern. Sie werden Turm-Objekte strategisch innerhalb der Spielarena einsetzen und objektorientierte Prinzipien wie Kapselung und Methodenaufruf anwenden, um die Erstellung robuster und wartbarer Spielmechaniken zu gewährleisten. Durch kollaborative Problemlösung werden die Schüler Herausforderungen bei der Gestaltung und Implementierung strategischer Interaktionen zwischen Türmen, Kugeln und Spielelementen angehen, praktische Einblicke in die Prinzipien des Spieldesigns gewinnen und ihr allgemeines Verständnis der Turmverteidigungs-Mechanik verbessern.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse vorausgesetzt, einschließlich Iteration und Programmverzeigungen. Die Schüler sollten bereits mit Greenfoot vertraut sein.

Dauer des Szenarios	1) Aufgabe Task 5.9 – Erstellen der Klassen Bullet (Geschoss, bzw. Kanonenkugel) und Tower (Turm, Verteidigungsturn) (10 Minuten) 2) Aufgabe Task 5.10 – Diskussion wie sich ein Bullet-Objekt bewegen soll und was passieren soll, wenn ein Enemy-Objekt getroffen oder der Rand der Arena erreicht wird. (10 Minuten) 3) Aufgabe Task 5.11 – Programmierung der Bewegung der Klasse Bullet (30 Minuten) 4) Aufgabe Task 5.12 - Diskussion wie ein Tower-Objekt ein Bullet-Objekt schießen soll (15 Minuten) 5) Aufgabe Task 5.13 - Diskussion wie ein Tower-Objekt beim Schießen mit den anderen Objekten interagieren soll (15 Minuten) 6) Aufgabe Task 5.14 – Programmierung des Schießens der Klasse Tower (30 Minuten) 7) Aufgabe Task 5.15 – Tower-Objekte in der Arena (20 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	In diesem 130-minütigen Unterrichtsszenario tauchen Schülerinnen und Schüler der Sekundarstufe in die Dynamik der Interaktionen von Turm und Kugel in der Greenfoot-Umgebung ein. Der Unterricht konzentriert sich auf die Entwicklung der Fähigkeiten der Schüler bei der Gestaltung und Implementierung strategischer Spielelemente, um ein ansprechendes und interaktives Spiel zu schaffen. Begonnen wird mit einer 10-minütigen Aufgabe, in der die Schüler die Bullet- und Tower-Klassen erstellen. Dieser Schritt legt den Grundstein für das Verständnis und die Umsetzung der Wechselwirkungen zwischen diesen Spielelementen. Im Anschluss daran folgt eine 10-minütige Diskussion darüber, wie sich Instanzen der Bullet-Klasse bewegen sollten und welche Aktionen ausgeführt werden sollten, wenn eine Kugel eine Instanz der Enemy-Klasse oder den Rand der Arena erreicht. Diese Diskussion schafft die Grundlage für die Implementierung präziser und dynamischer Bewegungsmechaniken. Die Schüler widmen dann 30 Minuten der Implementierung der Bewegung von Instanzen der Bullet-Klasse. Diese Aufgabe fordert die Schülerinnen und Schüler heraus, ihr Verständnis der Greenfoot-Methode move() und der Ereignisbehandlung anzuwenden, um ein realistisches Geschossverhalten in der

	Spielumgebung zu simulieren. Der Unterricht wird mit einer 15-minütigen Aufgabe fortgesetzt, in der herausgearbeitet wird, wie Instanzen der Tower-Klasse Instanzen der Bullet-Klasse abschießen sollten. In dieser Diskussion geht es um die Logik und die Bedingungen für das Auslösen von Kugelschüssen aus Türmen. Im Anschluss daran werden weitere 15 Minuten der Diskussion gewidmet, wie Instanzen der Tower-Klasse mit relevanten Objekten interagieren sollten, indem sie beim Schießen Nachrichten verwenden. In diesem Segment wird die Bedeutung der Objektkommunikation und des Auslösens von Ereignissen in der Spieleentwicklung hervorgehoben. Die Schülerinnen und Schüler verbringen dann 30 Minuten damit, den Schießmechanismus der Turmklasse zu implementieren. Diese Aufgabe verlangt von den Schülern, die Schießlogik in Objektinteraktionen zu überführen, um zu erreichen, dass Türme sinnvoll mit Feinden und anderen Spielelementen interagieren. Der Unterricht wird mit einer 20-minütigen Aufgabe abgeschlossen, die sich auf die Verwaltung und den Einsatz von Türmen in der Arena konzentriert. Diese Aufgabe umfasst die Platzierung, Interaktion und strategische Positionierung von Türmen, um die Spieldynamik zu optimieren und die Spieler herauszufordern. Als Ergebnis dieses Unterrichtsszenarios haben die Schüler praktische Erfahrungen in der Gestaltung und Umsetzung strategischer Interaktionen zwischen Türmen und Geschossen in Greenfoot gesammelt. Sie haben Fähigkeiten in objektorientierter Programmierung, Event-Handling und strategischem Spieldesign erworben, die sie darauf vorbereiten, ansprechende und dynamische Spiele zu erstellen.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

7.3.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 5.9 – Erstellen der Klassen Bullet (Geschoss bzw. Kanonenkugel) und Tower (Turm, Verteidigungsturm)

Gegenstand: Der Lehrer führt die Schüler durch die Erstellung der Bullet- und Tower-Klassen.

Zu besprechende Konzepte: Erstellung von Klassen, Rollen von Klassen und deren Ersteinrichtung.

Aktivität: Der Unterricht beginnt mit einer Wiederholung der Objekterstellung, der Programmierung von Objektebewegung und -interaktion innerhalb der Greenfoot-Umgebung. Die Lehrkraft führt mit den Schülerinnen und Schülern eine Diskussion, um die Rollen der verschiedenen Klassen und deren Zusammenhang im Spiel zu klären.

Die Schüler erstellen zwei neue Klassen in der Greenfoot-Umgebung. Es geht dabei um das Verständnis des Zwecks jeder einzelnen Klasse im Kontext des Spiels. Sie lernen, wie sie diese Klassen im Quellcode erzeugen und bereiten sich auf komplexere Interaktionen mit den Objekten vor.

Die Schüler beginnen mit der Erstellung der Bullet-Klasse. Diese Klasse repräsentiert Geschosse, die von den Türmen abgefeuert werden. Sie öffnen Greenfoot, wählen im Menü die neue Unterklasse aus der Actor-Klasse aus und nennen die neue Klasse “Bullet”.

Als Nächstes erstellen die Schülerinnen und Schüler die Tower-Klasse. Diese Klasse repräsentiert die Türme, die Geschosse auf Feinde abfeuern. Sie wählen erneut "Neue Klasse" aus dem Menü und nennen die neue Klasse “Tower”.

Zur Unterstützung erklärt der Lehrer die Rollen der Klassen Bullet und Tower innerhalb des Spiels. Die Bullet-Klasse steht für Geschosse, die von den Türmen abgefeuert werden, während die Tower-Klasse für stationäre Objekte steht, die Geschosse auf Feinde schießen können. Der Lehrer kümmert sich darum, dass die Schüler die unterschiedlichen Rollen verstehen, die jede Klasse spielt und wie sie im Spiel interagieren werden.

Am Ende dieser Aktivität sollten die Schülerinnen und Schüler grundlegende Strukturen für die Bullet- und Tower-Klassen erstellt und eingerichtet haben, um den Grundstein für eine detailliertere Umsetzung in zukünftigen Lektionen zu legen.

Commit*:* ece4df70042c8f60098e14ad2cee55514897d825

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/ece4df70042c8f60098e14ad2cee55514897d825

2. Aufgabe Task 5.10 - Disskussion wie sich ein Bullet-Objekt bewegen soll und was passieren soll, wenn ein Enemy-Objekt getroffen oder der Rand der Arena erreicht wird.

Gegenstand: Der Lehrer leitet eine Diskussion über das zu erwartende Verhalten eines Geschosses, während es sich im Spiel bewegt.

Zu besprechende Konzepte: Bewegungssteuerung, Kollisionserkennung.

Aktivität: Die Schülerinnen und Schüler führen ein Brainstorming aus und diskutieren die Regeln für die Bewegung des Geschosses und für die Kollisionsbehandlung.

Die Schülerinnen und Schüler besprechen zunächst, wie sich die Bullet-Instanz im Spiel bewegen soll. Sie sollten erkennen, dass sich ein Geschoss in einer geraden Linie in die Richtung bewegen sollte, in die es abgefeuert wurde, ohne die Richtung zu ändern. Die Geschwindigkeit des Geschosses sollte im Spiel verhältnismäßig gering und gleichmäßig sein. Um dies zu implementieren, können sie die eingebauten Bewegungsmethoden von Greenfoot verwenden. An dieser Stelle ist es sinnvoll, die Wirkungsweise des Konstruktors für die Initialisierung von Attributwerten beim Erstellen von Objektinstanzen hervorzuheben.

Der Lehrer lenkt die Diskussion dann darauf, was passieren soll, wenn das Geschoss den Rand der Welt erreicht oder mit einem Feind kollidiert. Die Schüler werden vermutlich vorschlagen, dass ein Geschoss aus dem Spiel verschwindet, wenn es den Rand der Welt erreicht. Sie diskutieren auch, dass bei einer Kollision von Geschoss und Feind das Geschoss verschwinden sollte und der Feind Schaden erleiden oder zerstört werden sollte.

Die Schülerinnen und Schüler schlagen den folgenden Pseudocode für die Bewegungs- und Kollisionslogik des Geschosses vor:

- Geschoss zuerst mit konstanter Geschwindigkeit einen Schritt vorwärtsbewegen und dann prüfen, ob das Geschoss den Rand der Welt erreicht hat.
- Falls wahr, entferne das Geschoss aus der Welt.
- Wenn nicht, dann ist zu prüfen, ob das Geschoss mit einem Feind kollidiert ist.
- Wenn es zu einer Kollision gekommen ist, dann bedeutet das, dass das Geschoss den Feind getroffen hat. In diesem Fall muss das Geschoss aus der Welt entfernt werden und dem Feind Schaden zufügen oder den Feind entfernen.

Der Lehrer demonstriert, wie diese Logik mit den Greenfoot-Methoden move(int), isAtEdge() und getOneIntersectingObject(Class cls) implementiert wird.

Der Lehrer ermutigt die Schüler, ihre Ideen zu verfeinern und über zusätzliche Details nachzudenken, wie z. B. die Anpassung der Geschwindigkeit an den Schwierigkeitsgrad des Spiels oder das Hinzufügen visueller Effekte, wenn ein Geschoss einen Feind trifft. Diese Schritte helfen den Schülern, die Prinzipien der Bewegungs- und Kollisionserkennung in der Spieleentwicklung zu verstehen, und bereitet sie auf die weitere Umsetzung in ihren Projekten vor.

3. Ausgabe Task 5.11 - Programmierung der Bewegung der Klasse Bullet

Gegenstand: Die Lehrkraft hilft den Schülerinnen und Schülern, die Bewegungssteuerung für die Bullet-Klasse zu implementieren.

Zu besprechende Konzepte: Bewegung von Actor-Objekten, Grenzen der Spielfeld-Welt.

Aktivität: Die Schüler schreiben Code, um das Geschoss vorwärts zu bewegen und es zu entfernen, wenn es den Rand der Arena erreicht. Dabei beginnen sie mit der Wiederholung der Actor-Bewegung und der Erkennung der Spielfeldgrenzen und konzentrieren sich darauf, wie diese Konzepte auf die Bullet-Klasse angewendet werden können. Die Schülerinnen und Schüler wissen aus vorherigen Aufgaben, wie sie der act()-Methode Code hinzufügen, um Interaktionen am Rand der Welt zu handhaben. Außerdem erinnern sie sich daran, wie man mit Richtungsänderungen umgeht, wenn ein Actor-Objekt in bestimmte Zellen eintritt. Es ist auch schon bekannt, wie man Zähler und Verzögerungsmechanismen in der act()-Methode für kontrollierte Bewegungen verwendet.

Mit diesem Wissen können die Schülerinnen und Schüler die Bewegungssteuerung für die Bullet-Klasse implementieren. Sie beginnen mit dem Hinzufügen der move(int)-Methode in der act()-Methode der Bullet-Klasse, um das Geschoss kontinuierlich vorwärts zu bewegen.

Commit*:* d372827a831381b2254f838041fa4d9a42e53b82

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/d372827a831381b2254f838041fa4d9a42e53b82

4. Aufgabe Task 5.12 - Diskussion wie ein Tower-Objekt Bullet-Objekte schießen soll

Gegenstand: Der Lehrer bespricht mit den Schülern wie die Geschosse aus dem Turm geschossen werden sollen.

Zu besprechende Konzepte: Objekterstellung, Methodenaufruf.

Aktivität: Die Schülerinnen und Schüler überlegen, wie der Turm Kugeln erzeugen und abfeuern wird. Die Schülerinnen und Schüler beginnen damit, die allgemeine Logik zu diskutieren, die ein Turm benötigt, um Kugeln im Spiel abzuschießen. Sie konzentrieren sich auf Schlüsselkonzepte wie das Erstellen von Bullet-Objekten und das Aufrufen von Methoden, wenn diese abgeschossen werden. Der Lehrer erinnert daran, dass der Turm nicht bei jedem Aufruf der act()-Methode Geschosse abschießen sollte. Das soll mit einer Verzögerungstechnik gehandhabt werden, ähnlich wie die Bewegung des Gegners.

Die Schülerinnen und Schüler überlegen sich, welche Schritte erforderlich sind, damit der Turm wiederkehrend Geschosse abfeuern kann. Der Lehrer regt an, einen Verzögerungsmechanismus für das Schießen zu verwenden. Dabei sollte erläutert werden, welche Rolle der Konstruktor bei der Implementierung dieses Mechanismus spielen könnte.

Der Lehrer erklärt, dass er ein neues Attribut, shootDelay, und einen Zähler, nextShootCounter in der Tower-Klasse einführen muss. Diese Attribute steuern die Schuss-Frequenz.

Die Lehrkraft schlüsselt die relevanten Schritte und Methoden für die Turm-Klasse auf. Sie definiert zunächst das shootDelay-Attribut, und initialisiert den nextShootCounter in der Tower-Klasse auf 0. Danach sollte die act()-Methode der Tower-Klasse modifiziert werden, um das Schießen zu handhaben. Die Methode sollte nur dann ein Bullet-Objekt erstellen und abschießen, wenn nextShootCounter 0 erreicht. Nach einem Schuss sollte nextShootCounter auf den Wert von shootDelay zurückgesetzt werden. Wenn nextShootCounter nicht 0 ist, sollte er um 1 dekrementiert werden. Am Ende sollte eine separate fire()-Methode definiert werden, um die Erstellung und den Start von Geschossen umzusetzen. Diese Methode instanziiert ein Bullet-Objekt und fügt es dem Spielfeld hinzu.

Durch diesen Schritt verstehen die Schüler, wie sie das Schießen implementieren können, indem sie die relevanten Schritte in Methoden der Tower-Klasse aufteilen. Der Lehrer unterstützt die Schüler, dass sie Methodenaufrufe und Objekterzeugung verstehen und ihr Verständnis für diese Techniken im Kontext ihres Spiels stärken.

5. Aufgabe Task 5.13 - Diskussion wie ein Tower-Objekt beim Schießen mit anderen Objekten interagieren soll

Gegenstand: Der Lehrer erklärt anhand von Nachrichten, wie der Turm mit Kugeln und anderen Objekten interagiert.

Zu besprechende Konzepte: Nachrichtenübergabe, Methodenaufrufe.

Aktivität: Die Schülerinnen und Schüler diskutieren das Prinzip zur Nachrichtenweitergabe beim Abschießen von Geschossen.

Der Teil beginnt damit, dass der Lehrer das Konzept der Nachrichtenweitergabe und seine Bedeutung in der objektorientierten Programmierung erklärt. Die Lehrkraft erklärt, wie Objekte im Spiel mithilfe von Methoden interagieren. Das kann man sich so erklären, dass die Objekte mittels Botschaften miteinander kommunizieren.

Um dies zu veranschaulichen, verwendet der Lehrer ein UML-Sequenzdiagramm, um die Interaktionen zwischen den Objekten Turm, Geschoss und Arena zu beschreiben. Das Diagramm stellt den Austausch von Botschaften und Methodenaufrufen visuell dar und hilft den Schülern, die Abfolge der Interaktionen zu verstehen.

Die Schülerinnen und Schüler besprechen den detaillierten Prozess, wie die Turmklasse beim Schießen mit den Geschossen und den anderen Objekten interagieren sollte. Der Lehrer erklärt, dass der Turm, wenn er sich entscheidet zu schießen, eine Botschaft sendet (Methodenaufruf), um ein Geschoss-Objekt zu erstellen und es in der Arena hinzuzufügen. Diese Interaktion wird innerhalb der act()-Methode der Tower-Klasse initiiert. Der Lehrer demonstriert anhand des UML-Sequenzdiagramms, wie der Tower eine Nachricht an das Greenfoot-Framework sendet, um der Welt ein neues Bullet-Objekt hinzuzufügen. Der Turm sendet dann eine Nachricht an die Bullet-Instanz und legt seine Richtung so fest, dass sie der aktuellen Drehung des Turms entspricht. Dadurch wird erreicht, dass sich das Geschoss in die vorgesehene Richtung bewegt. Sobald das

Geschoss erstellt und positioniert ist, interagiert es mit anderen Objekten im Spiel, beispielsweise mit den Feinden oder den Rändern der Welt. Der Lehrer erklärt, wie diese Interaktionen von der act()-Methode des Bullet-Objekts gehandhabt werden, die das Überprüfen auf Kollisionen und das Entfernen des Geschosses bei Bedarf beinhalten kann.

Der Lehrer hebt den kollaborativen Charakter dieser Interaktionen hervor und zeigt, wie der Algorithmus auf kooperierende Objekte verteilt ist. Dies hilft den Schülern, den modularen Aufbau und die klaren Kommunikationswege innerhalb ihres Spiels zu erkennen.

6. Aufgabe Task 5.14 - Programmierung des Schießens der Klasse Tower

Gegenstand: Die Lehrkraft führt die Schülerinnen und Schüler durch die Implementierung des Schießmechanismus für die Tower-Klasse.

Zu besprechende Konzepte: Objekterstellung, Positionierung von Actor-Objekten.

Aktivität: Die Schülerinnen und Schüler schreiben Code, damit der Turm Geschosse abfeuern kann.

Dieser Unterrichtsteil beginnt damit, dass der Lehrer das übergeordnete Ziel erklärt: den Schießmechanismus für die Turmklasse zu implementieren. Der Lehrer unterteilt die Aufgabe dann in überschaubare Schritte und führt die Schüler durch jeden einzelnen.

Zunächst bereiten die Schülerinnen und Schüler die notwendigen Attribute und Konstruktoren für die Tower-Klasse vor. Der Lehrer erklärt, dass der Turm ein Attribut braucht, um zu verwalten, wann er schießen kann. Als Nächstes erstellen die Schülerinnen und Schüler zwei Methoden: Boolean Tower.canShoot() und void Tower.fire(). Anfänglich können diese Methoden false bzw. nichts zurückgeben, sodass sie in der act()-Methode verwendet werden können. Die act()-Methode wird dann so aktualisiert, dass diese Methoden aufgerufen werden. Der Lehrer erklärt, dass die canShoot()-Methode true zurückgeben sollte, sobald der shootCounter 0 erreicht. Die fire()-Methode wird implementiert, um eine Bullet-Instanz zu erstellen und korrekt im Spielfeld zu positionieren.

Der Lehrer unterstützt die Schüler, jeden Teil des Codes verstehen, indem er die Erstellung des Bullet-Objekts zeigt, es in der Welt positioniert und seine Drehung mit dem Turm ausrichtet.

Die Schülerinnen und Schüler testen dann ihre Lösung, indem sie das Spiel ausführen, eine Turminstanz platzieren und überprüfen, ob sie in den entsprechenden Abständen Geschosse abschießt. Der Lehre fordert die Schüler auf, alle Probleme zu beheben und sicherzustellen, dass die Geschosse tatsächlich erstellt werden und sich wie erwartet bewegen.

Als Ergebnis haben die Schülerinnen und Schüler einen funktionierenden Schießmechanismus für die Turmklasse implementiert und dabei ihr Verständnis für die Objekterstellung, die Positionierung von Akteuren und den Methodenaufruf in Greenfoot gestärkt.

Commit*:* 62aec085954beacf996865a55bed312a09c675f2

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/62aec085954beacf996865a55bed312a09c675f2

7. Aufgabe Task 5.15 – Tower-Objekte in der Arena

Gegenstand: Mit der Hilfe des Lehrers bauen die Schüler die Türme und Geschosse in das Spiel ein, um eine funktionierende Spiel-Arena zu erhalten.

Zu besprechende Konzepte: Einbau von Objekten in das Spiel, Testen des Spiels.

Aktivität: Die Schüler platzieren Türme in der Arena und testen deren Interaktionen mit Geschossen und Feinden. Der Lehrer erklärt zunächst das Ziel: Türme in die Arena zu integrieren und zu erreichen, dass sie korrekt mit Geschossen und Feinden interagieren. In diesem Teil werden Türme in der Arena platziert und ihr Verhalten in der Spielumgebung getestet.

Die Schüler beginnen damit, Turm-Objekte an verschiedenen Positionen in der Arena zu platzieren. Der Lehrer erklärt, wie man Türme über die Greenfoot-Schnittstelle hinzufügt, um zu erreichen, dass jeder Turm richtig positioniert ist.

Als nächstes führt der Lehrer das Konzept des Überladens von Konstruktoren ein. Das ist hier nützlich für die Initialisierung von Turmobjekten mit unterschiedlichen Drehungen. Der Lehrer führt die Schüler dann durch das Erweitern der Tower-Klasse, um einen überladenen Konstruktor hinzuzufügen, der einen ganzzahligen Parameter für den Rotationswinkel entgegennimmt. Das ermöglicht die Platzierung und Ausrichtung der Türme innerhalb der Arena besser zu handhaben.

Commit*:* bfb6a271f490c341c760e654b3f86a87111c54cb

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/bfb6a271f490c341c760e654b3f86a87111c54cb

7.4. Objekte in der Greenfoot-Umgebung: Geschosse, Feinde und Spieldynamik

7.4.1. Dokumentation des Unterrichtsszenarios

Tabelle 17. Objekte in der Greenfoot-Umgebung: Geschosse, Feinde und Spieldynamik

Titel	Objekte in der Greenfoot-Umgebung: Geschosse, Feinde, und Spieldynamik
Lernziele	Die Schüler sollen Kenntnisse in der Gestaltung und Implementierung interaktiver Spieldynamiken mit den Klassen Bullet und Enemy in der Greenfoot-Umgebung erlangen. Darüber hinaus soll das Verständnis gestärkt werden, wie Objektinteraktionen funktionieren und dadurch eine Interaktion zwischen den Spielelementen entsteht. Die Teilnehmer demonstrieren dabei ihre Fähigkeit, eine präzise Kollisionserkennungs- und Reaktionsmechanismen zu implementieren, und zu beschreiben. Es geht insbesondere darum, wie Instanzen der Bullet-Klasse mit Instanzen der Enemy-Klasse interagieren. Die Schüler erhalten praktische Einblicke in wichtige Greenfoot-Methoden wie Greenfoot.showText(String, int), Greenfoot.getRandomNumber(int) und World.act(). Diese Methoden werden verwendet, um die Spielpräsentation zu verbessern, zufälliges Verhalten umzusetzen und Spielzustandsaktualisierungen zu verwalten. Darüber hinaus beherrschen die Schüler die Implementierung der Erzeugung von Feind-Objekten (Spawn) und die Umsetzung der Bedingungen, die zum Beenden des Spiels führen. Für den Spieler entsteht ein dynamisches Spielerlebnis. Durch eine Überarbeitung der Objektassoziationen festigen die Studierenden ihr Verständnis dafür, wie Objekte zusammenarbeiten, um eine ansprechende Spieldynamik zu erzeugen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse vorausgesetzt, einschließlich Iteration und Programmverzeigungen. Die Schüler sollten bereits mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Aufgabe Task 5.16 – Diskussion wie ein Bullet-Objekt mit anderen Objekten interagieren soll. (15 Minuten) 2) Aufgabe Task 5.17 – Programmierung des Verhaltens von Bullet-Objekten bei Berührung eines Enemy-Objekts (30 Minuten) 3) Erklärung der Funktionalität der Methoden Greenfoot.showText(String, int, int), Greenfoot.getRandomNumber(int) und World.act() (15 Minuten) 4) Aufgabe Task 5.18 – Erzeugen von Enemy-Objekten und Beenden des

	Spiels (30 Minuten) 5) Zusammenfassung über Klassen-Assoziation (20 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	In diesem 110-minütigen Unterrichtsszenario tauchen Schülerinnen und Schüler der Sekundarstufe in die Feinheiten der Spieldynamik mit Geschossen und Feinden in der Greenfoot-Umgebung ein. Der Unterricht konzentriert sich auf die Entwicklung der Fähigkeiten der Schüler bei der Erstellung eines interaktiven und dynamischen Spiels durch gut gestaltete Objektinteraktionen und Spielmechaniken. Der Unterricht beginnt mit einer 15-minütigen Diskussion darüber, wie Instanzen der Bullet-Klasse mithilfe von Botschaften mit anderen Objekten im Spiel interagieren sollten. Diese Diskussion bildet die Grundlage für das Verständnis, wie Objekte zusammenarbeiten, um bestimmte Spielverhaltensweisen zu erreichen. Anschließend widmen sich die Schüler 30 Minuten der Implementierung der Funktionalität, bei der Bullet-Objekte im Spiel auf Enemy-Objekte treffen. Diese Aufgabe erfordert von den Schülerinnen und Schülern, ihr Verständnis der Objektkollisionserkennung und der Ereignisbehandlung anzuwenden, um die Interaktionen zwischen den Objekten herzustellen. Das nächste Unterrichtsteil beinhaltet eine 15-minütige Erklärung der wichtigsten Greenfoot-Methoden: Greenfoot.showText(String, int, int), Greenfoot.getRandomNumber(int) und World.act(). Die Schüler erhalten Einblicke, wie diese Methoden zur Anzeige von Text, zur Generierung von Zufallszahlen für Spielmechaniken und zum Beeinflussen des Aktualisierungszyklus der Spielwelt beitragen. Die Schüler verbringen dann 30 Minuten mit der Programmierung der Erzeugung gegnerischer Objekte und den Bedingungen für das Beenden des Spiels. Dazu gehört das Entwerfen und Implementieren von Mechanismen, um Feinde in angemessenen Abständen erscheinen zu lassen, und die Festlegung von Bedingungen für das Beenden des Spiels auf der Grundlage von Spieleraktionen oder Spielzielen. Es folgt eine 20-minütige Wiederholungsphase, die sich auf die Vertiefung des Verständnisses von Objektassoziationen und der Anwendung bei der Implementierung der Spieldynamik konzentriert. Die Schüler werden ihr Verständnis dafür überprüfen und verfeinern, wie Objekte in der Greenfoot-

	Umgebung zusammenarbeiten, um die gewünschten Effekte im Spiel zu erzielen. Die Schüler haben im Ergebnis dieser Unterrichtseinheit ein tieferes Verständnis dafür entwickelt, wie man interaktive und ansprechende Spieldynamiken mit Geschossen, Feinden und strategischen Verfahren innerhalb von Greenfoot erstellt. Sie haben praktische Erfahrungen zur Implementierung von Objektinteraktionen, zur Verwaltung von Spielzuständen und zur Verbesserung der Spielerfahrung gesammelt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

7.4.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 5.16 - Diskussion wie ein Bullet-Objekt mit anderen Objekten interagieren soll.

Gegenstand: Der Lehrer moderiert eine Diskussion, wie Geschosse mit anderen Objekten, insbesondere Feinden, interagieren sollten, indem sie Botschaften weitergeben.

Zu besprechende Konzepte: Botschaften, Kollisionserkennung und Objektinteraktion.

Aktivität: Der Unterricht beginnt mit einem Überblick über Objektinteraktionen in der Greenfoot-Umgebung, wobei der Schwerpunkt darauf liegt, wie Instanzen verschiedener Klassen miteinander kommunizieren und sich gegenseitig beeinflussen. Die Lehrkraft regt eine Diskussion der Schülerinnen und Schüler an, um diese Konzepte und ihre praktischen Anwendungen in der Spieleentwicklung zu vertiefen.

Die Schülerinnen und Schüler führen ein Brainstorming durch und diskutieren die Interaktion der Geschosse mit den anderen Objekten und die Botschaften, die auszutauschen sind. Der Lehrer erklärt, wie Botschaften in der objektorientierten Programmierung zur Interaktion der Objekte benutzt werden. Im Speziellen geht es darum, wie Instanzen der Bullet-Klasse mit anderen Objekten wie Feinden und der Arena interagieren, insbesondere wenn ein Geschoss einen Feind trifft.

Die Schülerinnen und Schüler werden ermutigt, ihre Vorschläge zur Geschossinteraktionen auszutauschen. Sie beschäftigen sich mit Fragen wie:

- Was passiert, wenn ein Geschoss einen Feind trifft?
- Wie soll das Geschoss dieses Ereignis an andere Objekte kommunizieren?
- Welche Botschaften sind auszutauschen, um die Interaktion ordnungsgemäß nachzubilden?

Der Lehrer führt das Prinzip der Kollisionserkennung ein und erklärt, wie das Spiel erkennen muss, wenn ein Geschoss einen Feind trifft. Es werden auch die nachfolgenden Aktionen besprochen, wie z. B. die Absenkung der Spielstärke des Gegners oder das Entfernen des Gegners aus der Arena.

Um diese Interaktionen zu visualisieren, verwendet der Lehrer ein UML-Sequenzdiagramm. Das Diagramm veranschaulicht die Botschaften, die während der Interaktion zwischen den Klassen Bullet (Geschoss), Enemy (Feind) und Arena ausgetauscht werden.

2. Aufgabe Task 5.17 – Programmierung des Verhaltens von Bullet-Objekten bei Berührung eines Enemy-Objekts

Gegenstand: Der Lehrer führt die Schüler durch die Implementierung der Geschoss-Feind-Interaktion.

Zu besprechende Konzepte: Kollisionserkennung, Methodenaufruf und Änderungen des Objektzustands.

Aktivität: Die Schülerinnen und Schüler schreiben Code, um die Kollision zwischen einem Geschoss und einem Feind abzubilden, einschließlich der Auswirkungen der Kollision. Der Lehrer beginnt mit der Erläuterung des Prinzips der Kollisionserkennung und des Methodenaufrufs in Greenfoot. Die Schülerinnen und Schüler werden Schritt für Schritt angeleitet, um die Kollision zwischen Geschossen und Feinden zu implementieren.

Die Schülerinnen und Schüler bereiten zunächst die notwendigen Attribute und Methoden in den Klassen Bullet und Enemy vor. Danach wird Code in der Bullet-Klasse geschrieben, um eine Kollision mit einer Enemy-Instanz zu erkennen und die Enemy.hit(Bullet)-Methode aufzurufen. Die Schüler implementieren dann die Methode Enemy.hit(Bullet), um die Auswirkungen der Kollision abzubilden, beispielsweise um die Spielstärke des Feindes zu verringern oder ihn aus dem Spiel zu entfernen. Die Schüler sollen ihre Programme testen und überprüfen, dass die Interaktion zwischen dem Geschoss und dem Feind wie beabsichtigt funktioniert.

Als Ergebnis verfügen die Schüler jetzt über einen Kollisionserkennung zwischen Geschossen und Feinden mit geeigneten Methodenaufrufen und Objektzustands-änderungen. Diese Übung hat das Verständnis der Kollisionserkennung, der Methodenaufrufe und der praktischen Anwendung dieser Konzepte in der Spieleentwicklung gestärkt.

Commit*:* dcfe31bc006b7f3dcd8b8b759cc1be901c32913c

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/dcfe31bc006b7f3dcd8b8b759cc1be901c32913c

3. Erklärung der Funktionalität der Methoden Greenfoot.showText(String, int, int), Greenfoot.getRandomNumber(int) und World.act()

Gegenstand: Erklärung einiger spezifischer Greenfoot-Methoden, die im Spiel nützlich sind.

Zu besprechende Konzepte: Anzeige von Text, Erzeugen von Zufallszahlen, Verwendung der act()-Methode

Aktivität: Die Schülerinnen und Schüler lernen, wie man Text auf dem Bildschirm anzeigt, Zufallszahlen generiert und Verhalten des Spiels in der act-Methode implementiert.

Der Lehrer gibt eine Einführung in die Greenfoot-Methoden Greenfoot.showText(String, int, int), Greenfoot.getRandomNumber(int) und World.act(). Der Zweck dieser Methoden wird diskutiert und ihre Bedeutung für die Darstellung von Informationen, die Schaffung von Zufälligkeit und die Definition von Verhaltensweisen herausgearbeitet.

Die Greenfoot-Methoden showText(), getRandomNumber() und act() werden vom Lehrer vorgestellt.

Mit dem Aufruf der Methode Greenfoot.showText(String, int, int) wird Text im Spielfeld an angegebenen Koordinaten angezeigt. Diese Methode ist nützlich, um Spielinformationen wie Punktestände, die Spielstärke oder Anweisungen als Textnachricht anzuzeigen.

Die Methode Greenfoot.getRandomNumber(int) generiert eine Zufallszahl. Die als Argument angegebene Zahl n steuert, dass die Zufallszahl im Bereich von 0 bis n-1 liegt (einschließlich der angegebenen Werte). Diese Methode kann verwendet werden um Zufälligkeit in das Spiel einzuführen, z. B. das Erzeugen von Feind-Objekten an zufälligen Orten oder das Generieren zufälliger Bewegungsmuster.

Die Methode World.act() wird vom Greenfoot-Framework wiederholt aufgerufen, um die im Spiel auftretenden Ereignisse zu verarbeiten. Diese Methode ist der Ort, wo die Hauptaktionen und die Verfahren im Spiel untergebracht sind und damit kontinuierliche Aktualisierungen und Interaktionen innerhalb des Spiels ermöglicht werden.

4. Aufgabe Task 5.18 – Erzeugen von Enemy-Objekten und Beenden des Spiels

Gegenstand: Erzeugen von Feind-Objekten und die Umsetzung des Beendens des Spiels.

Zu besprechende Konzepte: Erzeugen von Objekten während des Spielverlaufs, Hauptschleife der Greenfoot-Anwendung und Ende des Spiels.

Aktivität: Die Schüler schreiben Code, um regelmäßig Feind-Objekte zu erzeugen und im Spiel zu platzieren. Darüber hinaus werden die Bedingungen definiert, um das Ende des Spiels auszulösen. Der Lehrer erklärt, dass es für die Spieldynamik wichtig ist, Feind-Objekte in regelmäßigen Abständen zu erzeugen und die Bedingungen für das Spielende zu definieren, wenn alle Feinde besiegt sind. Die Technik der dynamischen Objekterzeugungen, der Hauptschleife und der

Beendigung des Spiels werden diskutiert, um den Schülern ein klares Verständnis dafür zu vermitteln, was implementiert werden muss.

Die Methode Arena.act() wird verwendet, um periodisch Feind-Objekte zu erzeugen. Dafür sollte eine Verzögerungstechnik benutzt werden, um das Intervall zwischen dem Einsetzen von neuen Feind-Objekten zu steuern. Das wird in der spawn()-Methode in der Arena-Klasse realisiert. Diese Methode erstellt eine Instanz der Enemy-Klasse, weist dem Feind Eigenschaften zu (z. B. Position, Attribute) und fügt die Instanz zur Arena hinzu.

Die Schüler programmieren die Beendung des Spiels, wenn vorgegebene Bedingungen erfüllt sind. Das ist dann der Fall, wenn die Anzahl der Gegner (Enemy-Objekte) auf null sinkt. Dann ist das Spiel vorbei und der Spieler hat gewonnen. Um dies zu erreichen, sollten die Schülerinnen und Schüler ein Attribut in der Arenaklasse pflegen, um die Anzahl der erstellten Gegner zu verfolgen. Erhöht wird dieses Attribut jedes Mal, wenn ein Gegner-Objekt erzeugt wird, und verringert, wenn ein Gegner verschwindet. Die Methode Arena.kill(Enemy) sollte angepasst werden, um zu prüfen, ob alle Gegner besiegt wurden. Es soll eine Siegesnachricht angezeigt werden und das Spiel mit Greenfoot.stop() beendet werden. Der Aufruf von Greenfoot.stop() muss die letzte Anweisung in der Methode sein.

Zum Abschluss sollte das Verfahren zur dynamischen Erzeugung der Gegner-Objekte getestet werden, indem deren Erscheinen in der Arena beobachtet wird. Dabei ist zu prüfen, dass die Verzögerung zwischen den Objekt-Erzeugungen korrekt funktioniert. Auch das Ende des Spiels sollte geprüft werden, indem die Niederlage aller Gegner simuliert wird und geprüft wird, ob das Spiel mit einer Siegesnachricht gestoppt wird.

Als Ergebnis dieses Teilschritts haben die Schüler ein funktionierendes Verfahren zur Erzeugung von Gegner-Objekten implementiert und klare Bedingungen für das Spielende definiert. Das stärkt das Verständnis der Hauptschleife zur Spielsteuerung, des Objektmanagements und der Erzeugung von bedingungsabhängigen Spielergebnissen.

Commit*:* d48341a095561500af6032d5c8f56e201060f9a4

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/d48341a095561500af6032d5c8f56e201060f9a4

5. Zusammenfassung über Klassen-Interaktion

Gegenstand: Der Lehrer wiederholt rückblickend das Prinzip der Assoziationen zwischen Klassen und die Art und Weise, wie Objekte in Greenfoot interagieren und kommunizieren.

Zu besprechende Konzepte: Assoziation, Objektinteraktion und Botschaften.

Aktivität: Die Schülerinnen und Schüler frischen ihr Verständnis von Assoziationen auf, indem sie Verbindungen zwischen verschiedenen Objekten herstellen und deren Wechselwirkungen beschreiben.

Der Lehrer beginnt damit, das Thema Assoziationen zwischen Klassen noch einmal aufzugreifen. Dazu gehört, wie Objekte interagieren, kommunizieren und Botschaften aneinander weitergeben. Um dies zu erleichtern, kann der Lehrer auf Beispiele und Aufgaben zurückgreifen, die in früheren Unterrichtseinheiten behandelt wurden, um den Schülern zu helfen, ihr Wissen zu festigen und zu verstehen, wie diese Konzepte in der Greenfoot-Umgebung angewendet werden.

Dafür sollen UML-Sequenzdiagramme benutzt werden, um die Wechselwirkungen zwischen verschiedenen Objekten visuell darzustellen. Zum Beispiel kann man die Reihenfolge der Botschaften zeigen, wenn ein Geschoss einen Gegner trifft, oder zeigen wie die Arena das Erzeugen und Entfernen von Gegnern handhabt.

Am Ende sollen die Schüler ein vertieftes Verständnis für Assoziationen und Objektinteraktionen innerhalb der Greenfoot-Umgebung bekommen. Sie werden in der Lage sein, klar zu artikulieren, wie verschiedene Objekte in ihrem Spiel zusammenarbeiten, und diese Techniken auf ihre eigenen Spieleentwicklungsprojekte anzuwenden.

8. Vererbung

Für das Gebiet Vererbung werden vier Unterrichtsszenarien beschrieben.

8.1. Einführung in Klassen mit Vererbung

8.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 18. Einführung in Klassen mit Vererbung

Titel	Einführung in die Vererbung im Greenfoot-Umfeld
Lernziele	Durch dieses Unterrichtsszenario werden die Schüler in der Lage sein, die Konzepte der Vererbung zu verstehen. Die Technik der Vererbung zwischen Klassen wird im Kontext der Spieleentwicklung eingeführt. Daneben werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Grundlegendes zur Vererbung (15 Minuten) 2. Klassenhierarchie und Vererbung (15 Minuten) 3. Aufgaben Task 6.1 und 6.2: Identifikation gemeinsamer Eigenschaften (15 Minuten) und Identifikation der Basisklasse (15 Minuten) 4. Einführung in abstrakte Klassen (5 Minuten) 5. Aufgabe Task 6.3: Definition einer abstrakten Klasse im Spiel (10 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	In diesem 75-minütigen Unterrichtsszenario werden Schülerinnen und Schüler durch die Anwendung des Greenfoot-Tools in objektorientierte Programmierprinzipien mit Vererbung eingeführt. Der Unterricht beginnt mit einer 15-minütigen Einführung des Lehrers in grundlegende Vererbungskonzepte, die den Kontext zu früheren Sitzungen und der zukünftigen Spielentwicklung herstellt. Es folgt ein 15-minütiger Unterrichtsteil, in dem Schüler und Lehrer über die Klassenhierarchie innerhalb ihres Spiels diskutieren. Um vererbungsbezogene Konzepte zu erklären, werden die Klassen Orb und Direction betrachtet. In der

	nächsten 15-minütigen Aufgabe werden allgemeine Eigenschaften für diese Klassen untersucht. Es wird festgestellt, dass diese Klassen während ihrer Lebenszeit nicht aktiv agieren; sie reagieren ausschließlich Botschaften. Folglich wird eine gemeinsame Methode zum Handeln, die *act()*-Methode, identifiziert. Diese Methode wird in beiden Klassen mit einem leeren Quelltext definiert. Basierend auf der gefundenen gemeinsamen Eigenschaft wird in den nächsten 15 Minuten die neue Klasse PassiveActor implementiert, die eine Methode *act()* enthält. Im nächsten 5-Minuten-Abschnitt werden abstrakte Klassen eingeführt. Abstrakte Klassen dienen als Vorlagen für andere Klassen, aber können nicht selbst instanziiert werden. Sie sind jedoch für den Entwurf von Klassenhierarchien unerlässlich. Da es sich bei der PassiveActor-Klasse um eine Vorlage handelt, wird sie im nächsten 10-Minuten-Szenario als abstrakte Klasse definiert. PassiveActor ist Vorfahre der Klassen Orb und Direction, wodurch Orb und Direction ihre Nachkommen sind. Da die *act()*- Methode bereits in der PassiveActor-Klasse definiert ist, muss sie in den Klassen Orb und Direction nicht noch einmal programmiert werden. Die Schüler wurden im Ergebnis der Unterrichtseinheit mit Klassen und Vererbung vertraut gemacht.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Unter Berücksichtigung der Bedeutung der Vererbung für die objektorientierte Programmierung eröffnet die Projektstruktur Möglichkeiten für weitere Diskussionen und Erweiterung des Programms. In diesem Zusammenhang können weitere Klassen und deren Hierarchie betrachtet und zusätzliche Klassen, Methoden und Attribute eingeführt werden. Auf der anderen Seite kann der Lehrer dieses Thema adaptieren, um die Vorteile der Vererbung und der damit verbundenen Universalität nur auf den hier vorgeschlagenen Hierarchien aufzuzeigen.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

8.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Grundlegendes zur Vererbung

Gegenstand: Herstellung des Kontexts zu früheren Unterrichtseinheiten, Einführung und Erläuterung des Konzepts der Vererbung anhand von Beispielen aus der Praxis und Diskussion seiner Vorteile.

Zu besprechende Konzepte: Vererbung, Beispiele für Vererbung aus der Praxis.

Aktivität: Im Einführungsteil wird der Kontext zu den vorangegangenen Sitzungen hergestellt. Der Lehrer führt Vererbung ein. Der Lehrer sollte das Prinzip für die Schüler nachvollziehbar machen, indem er Beispiele aus dem wirklichen Leben verwendet (z. B. wenn die Eltern-Kind-Beziehung berücksichtigt wird, erben Kinder Eigenschaften von ihren Eltern, wie Haartyp, Augenfarbe usw.). Die Vorteile des Erbens im Kontext der Greenfoot-Umgebung und der Programmiersprache Java betrachtet.

2. Klassenhierarchie und Vererbung

Gegenstand: Einführung in die Klassenhierarchie bei Vererbung durch Erläuterung von Basisklassen und abgeleiteten Klassen. Es folgt eine Diskussion von Beispielen aus der Praxis und der Vererbung von Eigenschaften. Die Vorteile einer Klassenhierarchie sollen hervorgehoben werden.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Beispiele für Klassenhierarchien aus der Praxis, Vorteile der Vererbung und Klassenhierarchie.

Aktivität: Der Lehrer führt die Klassenhierarchie für Vererbung ein, dabei auch die Begriffe Basisklasse (auch bekannt als Superklasse, Elternklasse) und abgeleitete Klassen (auch bekannt als Unterklassen) ein:

- In diesem Zusammenhang können die bereits untersuchten realen Klassebeispiele besprochen werden
- Unterklassen erben Eigenschaften (d.h. Attribute und Methoden) von der Basisklasse
- Unterklassen können zusätzliche Eigenschaften enthalten, die in der Basisklasse nicht verfügbar sind.

Es wird erklärt, dass in der Programmiersprache Java jede Klasse mehrere Unterklassen haben kann, aber nur eine Basisklasse.

3. Aufgaben Task 6.1. und 6.2: Identifikation gemeinsamer Eigenschaften und Identifikation der Basisklasse

Gegenstand: Finden gemeinsamer Eigenschaften in den Klassen des Spiels, Bestimmen der Basisklasse und Implementierung einer neuen Klasse in der Klassenhierarchie.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Implementierung von Vererbung und Klassenhierarchie in der Spielentwicklung.

Aktivität: Die Klassen Orb und Direction werden betrachtet. Diese Klassen regieren auf Nachrichten. Deshalb sollte eine gemeinsame Methode zum Handeln, die *act()*-Methode, identifiziert werden. Auf Basis der identifizierten gemeinsamen Eigenschaften sollte eine neue Klasse PassiveActor implementiert werden, die eine Methode *act()* enthält:

- Die Klassen PassiveActor, Orb und Direction sollten zur Darstellung der Klassenhierarchie verwendet werden.
- Der Lehrer sollte die Klassenhierarchie mithilfe des Hierarchiediagramms illustrieren.

- Der Lehrer zeigt was sich in der Greenfoot-Umgebung geändert hat, wenn die Actor-Klasse durch PassiveActor ersetzt wurde

Commit*:* afe617814c07a5d885ed06479bf71deda8725f19

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/afe617814c07a5d885ed06479bf71deda8725f19

4. Einführung von abstrakten Klassen

Gegenstand: Einführung abstrakter Klassen, Erörterung ihrer Rolle als Vorlagen in Klassenhierarchien, Beispiele aus der Praxis, um die Anwendung abstrakter Klassen und deren Spezialisierung in Unterklassen zu veranschaulichen.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Abstrakte Klassen, Beispiele aus der Praxis für abstrakte Klassen im Kontext von Vererbung und Klassenhierarchie.

Aktivität: Abstrakte Klassen werden vom Lehrer eingeführt. Diese abstrakten Klassen dienen als Basisklassen für andere Klassen, aber können nicht instanziiert werden. Sie sind für das Entwerfen von Klassenhierarchien unerlässlich. Beispiele aus der Praxis, die sich auf abstrakte Klassen und Unterklassen beziehen, können vom Lehrer und den Schülern diskutiert werden (z. B. kann die Klasse Computer mit grundlegenden Eigenschaften als abstrakte Klasse definiert und auf Konsole, Desktop, Laptop und Mobiltelefon spezialisiert werden, jeweils mit einem bestimmten Satz von Eigenschaften usw.). Ein weiteres Beispiel sind geometrische Figuren. Die Klassen Rechteck und Dreieck können von der abstrakten Klasse Figur (im Sinne einer geometrischen Gestalt) erben. Bei der Berechnung des Umfangs und der Fläche der allgemeinen Figur kann noch keine Formel angegeben werden. Eine jeweils spezialisierte Formel wird in den Klassen Rechteck und Dreieck verwendet. Die Schülerinnen und Schüler sollten weitere Beispiele für geometrische Figuren und Körper diskutieren.

5. Aufgabe Task 6.3: Definition einer abstrakten Klasse im Spiel

Gegenstand: Rolle der PassiveActor-Klasse und deren Implementierung als abstrakte Klasse.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, abstrakte Klassen, Implementierung einer abstrakten Klasse.

Aktivität: Die Programmierung einer abstrakten Klasse wird in der Programmiersprache Java vorgenommen. Für das programmierte Greenfoot-Spiel ist die PassiveActor-Klasse eine Basisklasse, die act() umsetzt, aber noch keine weiteren speziellen Ausprägungen besitzt. Daher wird sie als abstrakte Klasse definiert und als Basisklasse der Klassen Orb und Direction eingetragen, wodurch Orb und Direction ihre abgeleiteten Klassen sind. Da die *act()*- Methode bereits in der PassiveActor-Klasse definiert ist, kann sie aus den Klassen Orb und Direction entfernt werden.

Commit*:* f7a5702cae29bf21c9c88620d01ef64e4127c21c

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/f7a5702cae29bf21c9c88620d01ef64e4127c21c

8.2. Konzepte der Vererbung (Teil 1)

8.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 19. Weiterführende Techniken der Vererbung

Titel	Weiterführende Techniken der Vererbung in Java
Lernziele	Durch dieses Unterrichtsszenario werden die Schüler in der Lage sein, weiterführende Techniken der Vererbung zu verstehen und anzuwenden. Die Techniken werden im Kontext der Spieleentwicklung behandelt. Daneben werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Aufgabe Task 6.4.: Finden gemeinsamer Eigenschaften bezüglich der Bewegung von Objekten im Spielfeld (15 Minuten) 2) Aufgabe Task 6.5.: Definition einer abstrakten Klasse für die Objektbewegung (15 Minuten) 3) Aufgabe Task 6.6.: Finden klassenspezifischer Eigenschaften (15 Minuten) 4) Einführung des Super-Schlüsselworts (20 Minuten) 5) Aufgabe Task 6.7.: Umgestaltung des Programmcodes im Zusammenhang mit Objekt-Bewegung (30 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 95-minütigen Unterrichtseinheit werden Schülerinnen und Schüler mit der Greenfoot-Umgebung in weiterführende Techniken im Zusammenhang mit Vererbung eingeführt. Der Unterricht beginnt mit einem 15-minütigen Abschnitt, in dem Eigenschaften im Zusammenhang mit Objektbewegungen untersucht werden, wobei der Fokus auf den Klassen Bullet und Enemy liegt. Diese Klassen verhalten sich während des Lebens ähnlich, sie bewegen sich auf die gleiche Weise im Spielfeld. Auf Basis der gefundenen gemeinsamen Eigenschaften wird in der nächsten 15-minütigen Aufgabe die neue abstrakte Klasse MovingActor implementiert, welche die Methode *act()* enthält. Diese Klasse ist eine

	gemeinsame Basisklasse, welche die Methode *act()* implementiert, um sich im Spielfeld zu bewegen. Die Unterklassen sollen sich auf ihren spezifischen Zweck konzentrieren. Darüber hinaus ist MovingActor die Basisklasse der Klassen Bullet und Enemy. Im nächsten 15-minütigen Abschnitt werden klassenspezifische Eigenschaften im Zusammenhang mit der Objektbewegung im Spielfeld untersucht. In diesem Zusammenhang wird die *act()*-Methode der jeweiligen Klassen untersucht, sowie die Attribute moveDelay und nextMoveCounter. Man stellt fest, dass der Code der *act()*-Methode, die für die Bewegung verantwortlich ist, derselbe ist. Es folgt ein 20-minütiger Unterrichtsteil, in dem das Super-Schlüsselwort im Rahmen der Vererbung eingeführt wird. Im letzten 30-Minuten-Abschnitt wird ein Code-Refactoring bezüglich der Objekt-Bewegung vorgenommen. Dies hat zur Folge, dass zuvor identifizierte Attribute *moveDelay* und *nextMoveCounter* aus den Unterklassen Bullet und Enemy in die Basisklasse MovingActor verschoben werden. Darüber hinaus ist der parametrische Konstruktor zum Initialisieren dieser Attribute in der MovingActor-Klasse definiert. Dieser Konstruktor mit geeigneten Parametern wird aus den Unterklassen Bullet und Enemy mit dem Schlüsselwort super aufgerufen. Darüber hinaus wird der Code, der für die Bewegung in der Methode *act()* der Unterklassen Bullet und Enemy verantwortlich ist, in die Methode *act()* der Klasse MovingActor verschoben. Der Rest der Implementierung in den Unterklassen bleibt unverändert. Schließlich wird die übergeordnete Version der Methode *act()* als erste Zeile der Methode *act()* in den Unterklassen Bullet und Enemy aufgerufen. Durch diese Unterrichtseinheit werden die Schüler mit weiterführenden Techniken der objektorientierten Programmierung, insbesondere mit Vererbung und mit abstrakten Basisklassen vertraut gemacht.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Unter Berücksichtigung der Bedeutung der Vererbung für die objektorientierte Programmierung eröffnet die Projektstruktur Möglichkeiten für weitere Diskussionen und Erweiterung des Programms. In diesem Zusammenhang können weitere Klassen und deren Hierarchie betrachtet und zusätzliche Klassen, Methoden und Attribute eingeführt werden. Auf der anderen Seite kann der Lehrer dieses Thema adaptieren, um die Vorteile der Vererbung und der damit verbundenen Universalität nur auf den hier vorgeschlagenen Hierarchien aufzuzeigen.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management

	System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

8.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 6.4.: Finden von gemeinsamen Eigenschaften bezüglich der Bewegung von Objekten im Spielfeld

Gegenstand: Untersuchung der Geschoss- und Feindklasse, Hervorhebung ähnlicher Verhaltensweisen, insbesondere wie sich die Objekte im Spielfeld bewegen und auf ihre Umgebung reagieren.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, abstrakte Klassen.

Aktivität: Der Fokus liegt auf den Klassen Bullet und Enemy, die sich ähnlich verhalten. Man stellt fest, dass sich diese Klassen auf die gleiche Weise bewegen und auf die Umgebung reagieren.

2. Aufgabe Task 6.5.: Definition einer abstrakten Klasse für die Objektbewegung

Gegenstand: Erstellen einer Basisklasse als abstrakte Klasse

Zu besprechende Konzepte: Basisklasse, abgeleitete Klassen, abstrakte Klassen.

Aktivität: Aufbauend auf den gefundenen gemeinsamen Eigenschaften wird die neue abstrakte Klasse MovingActor implementiert, die eine Methode *act()* enthält. MovingActor wird als Basisklasse der Klassen Bullet und Enemy entworfen, Bullet and Enemy sind abgeleitete Klassen. Es soll gezeigt werden, dass die abgeleiteten Klassen gemeinsame Eigenschaften von der Basisklasse erben. Die MovingActor-Klasse dient nur als Basisklasse für den Klassenentwurf und sollte daher als abstrakte Klasse deklariert werden.

Commit*:* 43e53b533563ce0a860b294ad9009f77409c48d4

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/43e53b533563ce0a860b294ad9009f77409c48d4

3. Aufgabe Task 6.6.: Finden klassenspezifischer Eigenschaften

Gegenstand: Untersuchung der Geschoss- und Feindklassen und Hervorhebung ihrer ähnlichen Verhaltensweisen bezüglich Objekt-Bewegung im Spielfeld.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, abstrakte Klassen.

Aktivität: Es werden klassenspezifische Eigenschaften bezüglich der Objektbewegung im Spielfeld untersucht. Die *act()*-Methode der jeweiligen Klassen wird untersucht, ebenso wie die Attribute

moveDelay und nextMoveCounter. Man stellt fest, dass der Code der *act()*- Methode, die für die Bewegung verantwortlich ist, derselbe ist.

4. Einführung des super-Schlüsselworts

Gegenstand: Einführung des super-Schlüsselworts im Kontext der Vererbung. Demonstration des super-Schlüsselworts für den Zugriff auf Basisklassenmethoden und Attribute der Basisklasse.

Zu besprechende Konzepte: Vererbung, super-Schlüsselwort und desssen Platzierung im Programmcode.

Aktivität: Das super-Schlüsselwort wird vom Lehrer eingeführt. Dieses Schlüsselwort kann im Rahmen der Vererbung für die folgenden Zwecke benutzt werden:

- um den Konstruktor aus der übergeordneten Klasse aufzurufen und Initialisierungsparameter zu übergeben,
- um eine Methode aus der übergeordneten Klasse aufzurufen,
- um auf ein Attribut aus der übergeordneten Klasse zuzugreifen.

Das Schlüsselwort *super* muss immer an der ersten Stelle des Ausdrucks stehen.

5. Aufgabe Task 6.7.: Umgestaltung des Programmcodes im Zusammenhang mit der Objekt-Bewegung

Gegenstand: Umgestaltung von Code in den Unterklassen Bullet und Enemy, und in der Basisklasse MovingActor.

Zu besprechende Konzepte: Vererbung, super-Schlüsselwort, Gestaltung des Programmcodes

Aktivität: Es wird eine Umgestaltung des Programmcodes (Code-Refactoring) vorgenommen. Die zuvor identifizierten Attribute *moveDelay* und *nextMoveCounter* aus den Unterklassen Bullet und Enemy werden in die Basisklasse MovingActor verschoben. Der parametrische Konstruktor zum Initialisieren dieser Attribute wird in der MovingActor-Klasse definiert. Dieser Konstruktor mit den notwendigen Parametern wird dann aus den Unterklassen Bullet und Enemy mit dem Schlüsselwort super aufgerufen. Der Code, der für die Bewegung in der Methode *act()* der Unterklassen Bullet und Enemy verantwortlich ist, wird in die Methode *act()* der Klasse MovingActor verschoben, während der restliche Programmcode in den Unterklassen unverändert bleibt. Schließlich wird die übergeordnete Version der Methode *act()* als erste Zeile der Methode *act()* in den Unterklassen Bullet und Enemy aufgerufen. Es sollte erwähnt werden, dass Unterklassen zusätzliche Eigenschaften enthalten können, die in der Basisklasse nicht verfügbar sind.

Commit*:* ca1f010a63445c1847b74259a1c6cd4817121db3

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/ca1f010a63445c1847b74259a1c6cd4817121db3

8.3. Konzepte der Vererbung (Teil 2)

8.3.1. Dokumentation des Unterrichtsszenarios

Tabelle 20. Fortgeschrittene Techniken der Vererbung

Titel	Fortgeschrittene Techniken der Vererbung in der Greenfoot-Umgebung
Lernziele	Am Ende dieser Unterrichtseinheit werden die Schüler fortgeschrittene Techniken im Kontext von Klassen und Vererbung verstehen. Daneben werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Aufgabe Task 6.8.: Erzeugen von benutzerdefinierten Feind-Objekten (30 Minuten) 2) Einführung in das Liskovsche Substitutionsprinzip (20 Minuten) 3) Aufgabe Task 6.9.: Dynamische Erzeugung von benutzerdefinierten Feind-Objekten (20 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 70-minütigen Unterrichtseinheit werden Schüler mit Greenfoot in fortgeschrittene Vererbungskonzepte eingeführt. Der Unterricht startet mit einem 30-minütigen Abschnitt, in dem die Schüler ausgehend von der Feindklasse zusätzliche Unterklassen erstellen, die verschiedene Feinde repräsentieren (beispielsweise Frosch und Spinne). In diesem Zusammenhang werden für jeden Gegnertyp Bilder und parameterlose Konstruktoren (mit entsprechendem Aufruf des übergeordneten Konstruktors) definiert. Im nächsten 20-Minuten-Abschnitt wird das Liskov-Substitutionsprinzip (LSP) eingeführt. Dieses Prinzip ist Teil der SOLID-Prinzipien des objektorientierten Designs. Das LSP besagt, dass Funktionen, die Verweise auf Objekte einer Klasse verwenden, in der Lage sein sollten, Objekte von Unterklassen zu verwenden. Im Anschluss daran wird eine 20-minütige Aufgabe bearbeitet, die dem dynamischen Erzeugen von benutzerdefinierten Feind-Objekten gewidmet ist. Die *Methode Arena.spawn()* wird betrachtet und benutzerdefinierte Feinde werden nach verschiedenen Kriterien erstellt und in einer Variablen vom Typ Enemy gespeichert. Es wird festgestellt, dass kein Code in der restlichen

	Anwendung geändert werden muss. Damit wird die Anwendung des LSP veranschaulicht. Ale Ergebnis wurden die Schüler in fortgeschrittene Vererbungskonzepte und praktische Anwendungen eingeführt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Unter Berücksichtigung der Bedeutung der Vererbung für die objektorientierte Programmierung eröffnet die Projektstruktur Möglichkeiten für weitere Diskussionen und Erweiterung des Programms. In diesem Zusammenhang können weitere Klassen und deren Hierarchie betrachtet und zusätzliche Klassen, Methoden und Attribute eingeführt werden. Auf der anderen Seite kann der Lehrer dieses Thema adaptieren, um die Vorteile der Vererbung und der damit verbundenen Universalität nur auf den hier vorgeschlagenen Hierarchien aufzuzeigen.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

8.3.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 6.8.: Erzeugen von benutzerdefinierten Feind-Objekten

Gegenstand: Definition mehrerer Unterklassen der Enemy-Klasse, jeweils mit Bildern und parameterlosen Konstruktoren, die den übergeordneten Konstruktor entsprechend aufrufen.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, super-Schlüsselwort zum Zugriff auf die Basisklasse

Aktivität: Ausgehend von der Klasse Enemy wird die Definition zusätzlicher Unterklassen vorgenommen, die verschiedene Feinde repräsentieren (z.B. Frosch und Spinne). Für jeden Gegnertyp sollten Bilder bereitgestellt werden und parameterlose Konstruktoren mit entsprechendem Aufruf des übergeordneten Konstruktors definiert werden.

Commit*:* b0ac1fbe793548a32f7700c292aed631918c8388

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/b0ac1fbe793548a32f7700c292aed631918c8388

2. Einführung in das Liskovsche Substitutionsprinzip

Gegenstand: Einführung des Liskov-Substitutionsprinzips, Vorteile der Einhaltung dieses Prinzips.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Referenzvariablen, Liskov-Substitutionsprinzip.

Aktivität: Das Liskov-Substitutionsprinzip wird eingeführt. Dieses Prinzip ist Teil der SOLID-Prinzipien des objektorientierten Designs. Das Prinzip besagt, dass Funktionen, die Verweise auf übergeordnete Klassen verwenden, in der Lage sein sollten, Objekte von Unterklassen zu verwenden. Um dieses Prinzip zu erklären, sollten Beispiele aus der Praxis diskutiert werden, z. B. wenn die Computer-Klasse als übergeordnete Klasse definiert ist und die Klassen Konsole, Desktop, Laptop und Mobiltelefon als Unterklassen definiert sind. Das Liskov-Substitutionsprinzip bedeutet, dass Funktionen, die die Computer-Klasse verwenden, auch mit allen Unterklassen funktionieren, ohne dass der Code geändert werden muss. Die Vorteile der Anwendung des Liskov-Substitutionsprinzips sollten diskutiert werden.

3. Aufgabe Task 6.9.: Dynamische Erzeugung von benutzerdefinierten Feind-Objekten

Gegenstand: Demonstration des Liskov-Substitutionsprinzips durch Erstellung von benutzerdefinierten Feind-Objekten.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Referenzvariablen, Liskov-Substitutionsprinzip.

Aktivität: Dieser Schritt ist der dynamischen Erzeugung von benutzerdefinierten Feind-Objekten gewidmet. Dazu wird die Methode *Arena.spawn()* ausgenutzt und benutzerdefinierte Feinde nach verschiedenen Kriterien erstellt und in einer Variablen vom Typ Enemy gespeichert.

Man stellt fest, dass dafür in der Anwendung kein weiterer Code geändert werden muss, was die Anwendung des Liskov-Substitutionsprinzips demonstriert.

Commit*:* 8cd4397f585ec957bbc18ca98e01823f434a13a6

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/8cd4397f585ec957bbc18ca98e01823f434a13a6

8.4. Konzepte der Vererbung (Teil 3)

8.4.1. Dokumentation des Unterrichtsszenarios

Tabelle 21. Anwendung von Klassen-Vererbung

Titel	Anwendung von Klassen-Vererbung
Lernziele	Als Ergebnis dieser Unterrichtseinheit werden die Schüler weitere Fähigkeiten zur Anwendung von Klassen und Vererbung erlangen. Die Anwendung erfolgt im Kontext der Spieleentwicklung. Daneben werden Kreativität, Teamwork und eine enthusiastische Herangehensweise an das Programmieren in der Greenfoot-Umgebung gefördert.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse und grundlegende Kenntnisse der objektorientierten Programmierung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1) Aufgabe Task 6.10.: Diskussion einer Klassenhierarchie von Arenen (20 Minuten) 2) Aufgaben Task 6.11 und 6.12.: Erzeugen einer Basisklasse Arena (30 Minuten) und einer spezialisierten Klasse DemoArena (15 Minuten) 3) Aufgabe Task 6.13.: Erzeugen benutzerspezifischer Arenen (30 Minuten) 4) Rückblick auf Techniken der Vererbung (20 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	Während dieser 115-minütigen Unterrichtseinheit wenden Schülerinnen und Schüler fortgeschrittene Vererbungstechniken an. Der Unterricht beginnt mit einer 20-minütigen Diskussion über die Arena-Klassenhierarchie. Unterklassen einer allgemeinen Arena-Basisklasse sind für benutzerdefinierte Layouts verantwortlich (z. B. Positionen von Orb- und Direction-Objekten, Größe der Arena). Diese Aufgaben werden in den Konstruktoren der Unterklassen ausgeführt, die die Position und Rotation der Objekte in der Arena und die Dimension der jeweiligen Arena festlegen und speichern. In der anschließenden 30-minütigen Aufgabe wird eine universelle Arena-Basisklasse eingeführt. Zusätzliche Attribute (*spawnPositionX*, *spawnPositionY* und *spawnRotation*) werden definiert, im Konstruktor initialisiert und in den Methoden *spawn()* und *respawn(Enemy)* verwendet. Attribute, die sich auf die

	Abmessungen der Arena (*Breite* und *Höhe*) beziehen, werden ebenfalls im Konstruktor definiert und initialisiert. Da die Arena-Basisklasse ausschließlich als Vorlage für die Definition konkreter Arenen dient, wird sie als abstrakte Klasse definiert. Aufbauend auf der Arena-Basisklasse wird in der nächsten 15-minütigen Aufgabe die DemoArena-Unterklasse eingeführt. Der DemoArena-Konstruktor wird definiert, wobei der Konstruktor der Basisklasse aufgerufen wird, und der Code, der für das Layout von Richtungen, Geschossen und Türmen verantwortlich ist, wird vom Arena-Konstruktor in den DemoArena-Konstruktor verschoben. Schließlich wird eine neue Instanz der DemoArena-Klasse erstellt. Im danach folgenden 30-minütigen Abschnitt werden weitere Unterklassen der Basisklasse Arena erstellt. Der Code kann mit anderen Schülern in der Gruppe geteilt werden. Die letzten 20 Minuten werden für einen Rückblick auf die Technik der Vererbung und deren Anwendung genutzt. Die Schüler haben dann fortgeschrittene Vererbungskonzepte und praktische Anwendungen kennengelernt.
Bewertung	Gamification erlaubt nur eine nicht-formale Bewertung, wird aber das Interesse, die intrinsische Motivation und den Lernoutput der gesamten Gruppe steigern. Unter Berücksichtigung der Bedeutung der Vererbung für die objektorientierte Programmierung eröffnet die Projektstruktur Möglichkeiten für weitere Diskussionen und Erweiterung des Programms. In diesem Zusammenhang können weitere Klassen und deren Hierarchie betrachtet und zusätzliche Klassen, Methoden und Attribute eingeführt werden. Auf der anderen Seite kann der Lehrer dieses Thema adaptieren, um die Vorteile der Vererbung und der damit verbundenen Universalität nur auf den hier vorgeschlagenen Hierarchien aufzuzeigen.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

8.4.2. Leitfaden zur Unterrichtsvorbereitung

1. Aufgabe Task 6.10.: Diskussion einer Klassenhierarchie von Arenen

Gegenstand: Vorschlag und Diskussion einer Arena-Klassenhierarchie. Bestimmen, wie Unterklassen für die Definition benutzerdefinierter Layouts verantwortlich sind, und Implementieren dieser Layouts in ihren jeweiligen Konstruktoren.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Konstruktoren.

Aktivität: Die Arena-Klassenhierarchie wird erstellt. Diese wird so konstruiert, dass die Unterklassen von Arena für benutzerdefinierte Layouts verantwortlich sind (z.B. Positionen von Orb- und Direction-Instanzen, Größe der Arena). Diese Aufgaben werden in den Konstruktoren der Unterklassen ausgeführt, die die Positionen und Rotationen der erzeugten Objekte, sowie die Dimension der Arena festlegen und speichern.

2. Aufgaben Task 6.11. und 6.12.: Erzeugen einer Basisklasse Arena und einer spezialisierten Klasse DemoArena

Gegenstand: Einführung einer universellen abstrakten Arena-Basisklasse, Definition und Initialisierung einer konkreten Arena-Unterklasse, Betrachtung der Klassenhierarchie.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Abstrakte Klassen, Konstruktoren.

Aktivität: Auf Basis der vorangegangenen Diskussion wird eine universelle Arenaklasse eingeführt. Zusätzliche Attribute (*spawnPositionX*, *spawnPositionY* und *spawnRotation*) werden definiert, im Konstruktor initialisiert und in den Methoden *spawn()* und *respawn(Enemy)* verwendet. Attribute, die sich auf die Abmessungen der Arena (*Breite* und *Höhe) beziehen,* werden ebenfalls im Konstruktor definiert und initialisiert. Da die Arena-Klasse als Vorlage für die Definition konkreter Arenen dient, wird sie als abstrakte Klasse definiert.

Commit*:* e9844d7d9b5f19969618b469ebc907d0fe3c1357

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/e9844d7d9b5f19969618b469ebc907d0fe3c1357

Aufbauend auf die Arena-Basisklasse wird die abgeleitete Unterklasse DemoArena definiert. Im DemoArena-Konstruktor wird der Konstruktor der übergeordneten Klasse aufgerufen. Damit kann der Code, der für das Layout von Direction-Objekten, Bullet-Objekten und Tower-Objekten verantwortlich ist, vom Arena-Konstruktor in den DemoArena-Konstruktor verschoben werden. Schließlich wird eine neue Instanz der DemoArena-Klasse erstellt. Um DemoArena zu aktivieren, klicken Sie mit der rechten Maustaste und wählen Sie neue DemoArena.

Commit*:* 6a6569774b5735f453a56c7cb2cdbf19d228eae9

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/6a6569774b5735f453a56c7cb2cdbf19d228eae9

3. Aufgabe Task 6.13.: Erzeugen benutzerspezifischer Arenen

Gegenstand: Definieren und Initialisieren von benutzerdefinierten Arena-Unterklassen, Untersuchen der Arena-Klassenhierarchie.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, abstrakte Klassen, Konstruktoren.

Aktivität: Weitere Unterklassen der Klasse Arena werden erstellt. Der Code kann mit anderen Schülern in der Gruppe geteilt werden.

4. Rückblick auf Techniken der Vererbung

Gegenstand: Wiederholung der Vererbung unter Java-Klassen, Herausarbeiten der Vorteile der Vererbung, Wiederholung von Klassenhierarchie und abstrakten Klassen, Wiederholung des super-Schlüsselworts, Wiederholung des Liskov-Substitutionsprinzips, Darstellung von Vererbungsbeispielen und -implementierungen aus der Praxis und im Spiel.

Zu besprechende Konzepte: Vererbung, Klassenhierarchie, Abstrakte Klassen, das Schlüsselwort super, das Liskov-Substitutionsprinzip, Beispiele und Implementierungen für reale und spielbezogene Vererbung.

Aktivität: Das Konzept der Vererbung und die dadurch entstehenden Vorteile werden wiederholt. Die Klassenhierarchie wird dazu noch einmal dargestellt und dabei auch die Rolle von abstrakten Klassen. Das super-Schlüsselwort zum Aufruf von Basisklassenmethoden und dem Zugriff auf Basisklassenattribute wird wiederholend erklärt. Auch das Liskov-Substitutionsprinzip wird noch einmal kurz erklärt, um über die Referenz einer Basisklasse auf jede mögliche Unterklasse zugreifen zu können. Es werden spielbezogene Vererbungsbeispiele und deren Umsetzung diskutiert.

9. Kapselung von Eigenschaften und Verhalten

Im Rahmen der Themeneinheit Kapselung wurden zwei Unterrichtsszenarien erstellt.

9.1. Kennenlernen der Kapselung

9.1.1. Dokumentation des Unterrichtsszenarios

Tabelle 22. Weiterentwicklung des Spiels mit Fokus auf Kapselung (Unterrichtsszenario 1)

Titel	Kennenlernen der Kapselung durch Spieleentwicklung mit Greenfoot
Lernziele	Der Zweck dieses Unterrichtsszenarios ist, den Schülern durch die Weiterentwicklung des TowerDefense-Spiels die Kapselung näher zu bringen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse, grundlegende Kenntnisse der objektorientierten Programmierung und Vererbung vorausgesetzt. Die Schüler sollten mit Greenfoot vertraut sein.
Dauer des Szenarios	1. Einführung (5 Minuten) 2. Aufgabe Task 7.1.: Zusammenarbeit im Team und Kodierung (20 Minuten) 3. Aufgaben Task 7.2 und 7.3.: Zusammenarbeit im Team (30 Minuten) 4. Diskussion (35 Minuten) 5. Erklärung des Codes (25 Minuten) 6. Aufgabe Task 7.4.: Teambildung und Projektzuweisung (10 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projektquellcode aus Github/Gitlab. Internet-Ressourcen.
Beschreibung	In dieser Unterrichtseinheit lernen Schülerinnen und Schüler der Sekundarstufe etwas über die Kapselung im Rahmen der objektorientierten Programmierung. Der Unterricht beginnt mit einer kurzen 5-minütigen Einführung, in der der Lehrer die Ziele umreißt. Die Schüler beginnen mit der Entwicklung der ManualTower-Klasse als Unterklasse der Tower-Klasse. Diese Arbeit konzentriert sich auf das Definieren von zwei Konstruktoren, die mit den Konstruktoren der übergeordneten Klasse konsistent sind und so eine ordnungsgemäße Initialisierung sicherstellen. Sie sollen auch eine act()-Methode implementieren, die zuerst die act()-Methode der übergeordneten Klasse aufruft. Nach der Erstellung der Klasse führt der Lehrer ein boolesches Attribut

	isManualControlled ein, das mit false initialisiert ist. Die Schüler erstellen eine Methode changeControl(boolean), die den isManualControlled-Zustand umschaltet und das Bild des Turms entsprechend ändert. Damit wird die Kapselung demonstriert, indem der Zugriff auf den Zustand eines Objekts über Methoden gesteuert wird. Jeder Schüler sollte dann die changeControl()-Methode manuell auf Instanzen von ManualTower auslösen und beobachten, wie sich der interne Zustand und die externe Repräsentation ändern. Kern des Unterrichts ist die Entwicklung einer privaten Methode processUserControl(), die Mausklicks auf die Tower-Instanz erkennen soll. Wenn man darauf klickt, ändert die Methode den Steuerungsstatus des Turms und aktualisiert seine Ausrichtung basierend auf der Mausposition. Kapselung wird verwendet, um die komplexe Steuerungslogik auszublenden. Die Schüler sollten die Methode implementieren und in die act()-Methode integrieren, die Interaktion mit der Spielumgebung testen, um die Funktionalität herzustellen, und um zu lernen, wie private Methoden den Objektzustand vor externen Änderungen schützen.
Bewertung	Diese Aktivität ermöglicht es den Lehrern, auf Basis der Diskussionen, der Überwachung des Flipped Classroom und der Teamarbeit der Schüler formatives Feedback zur Bewertung zu geben. Die Peer-Review-Bewertung wird online im Rahmen einer Hausaufgabe durchgeführt. Dies erinnert die Schüler an wichtige Aspekte der Übung, veranlasst sie, die Arbeit anderer Schüler kritisch zu bewerten, gibt ihnen Einblicke in gute oder weniger gute Lösungen ihrer Mitschüler usw. und verbessert das Gesamtergebnis der Lernergebnisse. Auch die Arbeit im Teamprojekt, an der die Schülerinnen und Schüler arbeiten, wird von diesen Lernergebnissen und Kenntnissen profitieren.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Schüler können die Diskussion zum Thema in dem Forum fortsetzen, das ihnen über das Lernmanagement-Tool zur Verfügung gestellt wird.

9.1.2. Leitfaden zur Unterrichtsvorbereitung

1. Einführung

Der Lehrer soll das zuvor entwickelte Spiel starten und mit den Schülern beobachten, wie sich verschiedene Actor-Objekte verhalten. Dabei wird der Vorschlag gemacht, eine andere Variante eines Turms zu entwickeln, der manuell gesteuert werden kann, um Feinde schneller zu entfernen. Der Spieler sollte in der Lage sein, jeweils einen Turm zu steuern. Wenn der Turm angeklickt wird, sollte er manuell gesteuert werden. Um anzuzeigen, welcher Turm manuell gesteuert wird, sollte der aktuell gesteuerte Turm ein anderes Aussehen haben.

2. Aufgabe Task 7.1.: Zusammenarbeit im Team und Kodierung

Gegenstand: Vorbereitung des Klasse ManualTower

Zu besprechende Konzepte: Vererbung, Klassen, Konstruktoren

Aktivität: Da die Schüler bereits wissen, wie man eine abgeleitete Klasse erstellt, können Teams gebildet werden, die jeweils eine ManualTower-Klasse als Nachkomme der Tower-Klasse erstellen. Die Teilnehmer sollten sowohl die Konstruktoren als auch die act()-Methode implementieren und darauf achten, dass die Basisklassenkonstruktoren in den Konstruktoren aufgerufen werden. In diesem Teil des Unterrichts wiederholen die Schülerinnen und Schüler den bisherigen Stoff, wenden ihn an und verbessern ihr praktisches Wissen über die Vererbung.

Commit*:* 63a02fa0c5080165cba8b467da08c4b65f31d0a8

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/63a02fa0c5080165cba8b467da08c4b65f31d0a8

3. Aufgaben Task 7.2 und Task 7.3: Zusammenarbeit im Team

Gegenstand: Notwendigkeit privater Methoden, am Beispiel der Methode changeControl()

Zu besprechende Konzepte: Methoden, Klassen, Attribute, Modifizierer für Sichtbarkeit

Aktivität: Der Lehrer sollte Symbole für den manuell gesteuerten Turm vorbereiten. Um das Symbol des Objekts programmgesteuert zu ändern, sollte der Lehrer den Schülern erklären, wie die Methode Actor.setImage(String) verwendet wird. Den Schülern wird etwas Zeit gegeben, um diese Funktion zu testen.

Die Lehrkraft bespricht mit den Schülerinnen und Schülern, wie festgestellt werden kann, ob der Turm manuell gesteuert wird. Dabei ist es nicht nur wichtig, den Zustand des Objekts zu ändern, sondern auch sein Bild zu aktualisieren. Wenn ein Benutzer den Status eines Turmobjekts ändern möchte und nur das Attribut direkt ändert, dann ändert sich das Bild nicht. Eine Diskussion soll den Schülerinnen und Schülern helfen zu verstehen, dass es notwendig ist, den Wert eines Attributs und das damit verbundene Bild durch eine Methode zu ändern und Attribute privat statt öffentlich

zu halten. Diese Praxis wird als Kapselung bezeichnet, bei der der interne Zustand verborgen ist und öffentliche Methoden verwendet werden, um diesen Zustand kontrolliert zu ändern.

Die Schülerinnen und Schüler sollen eine dementsprechende Methode changeControl() implementieren. Die Methode ist manuell aufzurufen und Änderungen im internen Zustand zu beobachten.

Commit*:* 2257746b7dac5eaab7acc55d6493319230338f3a

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/2257746b7dac5eaab7acc55d6493319230338f3a

4. Diskussion

Gegenstand: Grundlegendes zur Kapselung von Anweisungen in einer separaten Methode

Zu besprechende Konzepte: Methoden, Verzweigung

Aktivität: Der Lehrer weist darauf hin, dass der Zustand des Turms bisher nur durch manuelles Aufrufen der Methode changeControl() geändert werden kann. Das Ziel ist, die Methode im Spiel durch einen Mausklick ausgelöst aufzurufen. Dabei müssen einige Umstände beachtet werden. Es ist möglich, dass sich die Maus außerhalb der Welt befindet, in diesem Fall sind die Mausinformationen null. Die Schüler sollen auch daran erinnert werden, dass die act()-Methode während des Spiels ständig ausgeführt wird und dass man überprüfen sollte, ob das Objekt angeklickt wurde, und erst dann die changeControl()-Methode aufrufen sollte. Hervorzuheben ist, dass die Verarbeitung des Steuerelements in einer separaten Methode processUserControl() gekapselt werden sollte.

5. Erklärung des Codes

Gegenstand: Einführung einer Methode zur Lösung des Problems

Zu besprechende Konzepte: Methoden, Greenfoot-Umgebung

Aktivität: Es wird überlegt, wie man den Status eines Actor-Objekts ändern kann, wenn auf das Objekt geklickt wurde. Dafür kann die Methode GreenFoot.mouseClicked(Object) benutzt werden. Außerdem wird das MouseInfo-Objekt benötigt, das zum Abrufen von Informationen über die Mausposition verwendet werden kann.

6. Aufgabe Task 7.4.: Teambildung und Projektzuweisung

Gegenstand: Verständnis für private Methoden und Kapselung durch praktische Aufgaben

Zu besprechende Konzepte: Klassen, Methoden, Zugriffs-Modifikatoren

Aktivität: Nachdem die private Methode processUserControl() definiert wurde, sollen die Schüler die nötigen Schritte implementieren. Wenn mit der Maus geklickt wird, sollte sich der gesteuerte Turm ändern. Wenn der Turm manuell gesteuert wird, sollte er folgen und auf die Maus gerichtet sein. Erinnern Sie sie daran, dass es möglich ist, dass sich die Maus außerhalb der Welt befindet.

Nachdem Sie die Schüler in Teams eingeteilt haben, lassen Sie sie die processUserControl()-Methode implementieren.

Ein oder zwei Teams sollen im Anschluss ihre Arbeit vorstellen und die Ergebnisse mit dem Lehrer diskutieren. Als Ergebnis sollten alle Schüler verstanden haben, wie diese Methode umgesetzt wurde.

Commit*:* 6ec1f489576019a6493490f9e97797920b923869

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/6ec1f489576019a6493490f9e97797920b923869

9.2. Erlernen der Datenkapselung durch Spieleentwicklung mit Greenfoot

9.2.1. Dokumentation des Unterrichtsszenarios

Tabelle 23. Weiterentwicklung des Spiels mit Fokus auf Kapselung (Unterrichtsszenario 2)

Titel	Erlernen der Datenkapselung durch Spieleentwicklung mit Greenfoot
Lernziele	Der Zweck dieses Unterrichtsszenarios ist es, den Schülern durch die Weiterentwicklung des TowerDefense-Spiels die Kapselung näher zu bringen.
Zielgruppe	Schülerinnen und Schüler der Sekundarstufe, die am OOP4Fun-Kurs teilnehmen. Es werden grundlegende Programmierkenntnisse einschließlich Vererbung vorausgesetzt. Die Schüler sollten mit Greenfoot im Allgemeinen vertraut sein.
Dauer des Szenarios	1) Flipped Classroom Session (30 Minuten): Die Schülerinnen und Schüler sollten das Problem mit der zuvor implementierten Benutzersteuerung identifizieren und vorschlagen, wie das Problem gelöst werden kann. 2) Klassen-Attribute (5 Minuten) 3) Task 7.6.: Hinzufügen eines Verweises für einen manuell gesteuerten Turm (5 Minuten) 4) Methode der Klasse (10 Minuten) 5) Aufgabe Task 7.7.: Auswahl des manuell gesteuerten Turms von einer zentralen Stelle (20 Minuten) 6) Task 7.8.: Wechsel des manuell gesteuerten Turms aufrufen (15 Minuten) 7) Wiederholung (10 Minuten)
Materialien und Ressourcen	Das Lehrbuch aus dem Projekt OOP4Fun. Ressourcen aus dem OOP4Fun-Projekt. Projekt-Quellcode von Github/Gitlab. Internet-Ressourcen.
Beschreibung	Die Sitzung beginnt damit, dass die Schüler Probleme der Benutzersteuerung analysieren, beispielsweise die fehlende Möglichkeit, einen Turm abzuwählen, sobald er ausgewählt ist. Die Teilnehmer sollten in die Diskussion einbezogen werden und Lösungen vorschlagen, um den aktuell gesteuerten Turm zu nachzuverfolgen und die ManualTower-Klasse so zu modifizieren, dass sie eine Möglichkeit zum Abwählen des Turms bekommt. Schließlich sollte eine Klassenmethode changeControlledInstance() implementiert werden, die es ermöglicht, die Steuerung von Türmen von einer zentralisierten Methode aus zu ändern. Dabei wird auch gezeigt, wie Klassenmethoden den gemeinsamen Zustand über Instanzen hinweg verwalten können. Das Konzept der Kapselung wird mit einem umfassenden Bildungsansatz gelehrt und seine Bedeutung und Nützlichkeit in realen Anwendungen gezeigt.

	In dieser Unterrichtsszenario werden auch die Problemlösungs- und Kooperationsfähigkeiten unter den Schülern entwickelt.
Bewertung	Diese Aktivität ermöglicht es den Lehrern, auf der Grundlage der Diskussionen und der Überwachung des Flipped Classroom der Schüler Bewertungsfeedback zu geben. Auch die Arbeit im Teamprojekt, an der die Schülerinnen und Schüler arbeiten, wird von diesen Lernergebnissen und Kenntnissen profitieren.
Bereitstellung der Ergebnisse	Um ihre Ergebnisse an Lehrende und Mitschüler weiterzugeben, wird die übliche Einrichtung von Github/Gitlab und ein Learning Management System (z.B. Moodle) verwendet. Die Studierenden können die Diskussion zum Thema in dem ihnen zur Verfügung gestellten Forum über das Lernmanagement-Tool fortsetzen.

9.2.2. Leitfaden zur Unterrichtsvorbereitung

1. Flipped Classroom Session

Gegenstand: Notwendigkeit eines gemeinsamen Attributs für Objekte einer Klasse

Zu besprechende Konzepte: Statische Klassenattribute (static, gemeinsame Attribute aller Objekte einer Klasse)

Aktivität: Die Schülerinnen und Schüler sollen zu Beginn des Unterrichts ein Problem identifizieren. Derzeit ist es nicht möglich, die Auswahl des gesteuerten Turms aufzuheben. Die Schüler sollen darüber nachdenken, wie dieses Problem gelöst werden könnte. Der Ausgangspunkt ist, dass im Spiel jeweils nur ein Turm für manuelle Steuerung ausgewählt werden sollte. Diese Diskussion sollte die Schüler auf die Idee führen, dass es einen Ort im Programm gibt, der nur einmal initialisiert wird und auf den von anderen Teilen des Programms, von anderen Objekten und Akteuren aus zugegriffen werden kann.

2. Klassen-Attribute

Gegenstand: Einführung in die Grundlagen von Klassenattributen

Zu besprechende Konzepte: Klassen, Attribute, Statische Attribute

Aktivität: Es wird erklärt, was Klassenattribute sind: Variablen, die zur Klasse selbst und nicht zu Instanzen der Klasse gehören. Dieses Konzept kann auf das zuvor besprochene Spielszenario angewendet werden, indem ein zentrales Attribut zur Verwaltung des aktuell ausgewählten Turms benutzt wird.

3. Aufgabe Task 7.6.: Hinzufügen eines Verweises auf einen manuell gesteuerten Turm

Gegenstand: Praktische Verwendung von Klassenattributen und null-Wert

Zu besprechende Konzepte: Klassen, Attribute, Klassenattribute, Zugriffsmodifizierer

Aktivität: Um nachzuverfolgen, welcher Turm derzeit im Spiel für die manuelle Steuerung ausgewählt ist, wird der ManualTower-Klasse ein private static Attribut controlledInstance hinzugefügt und auf null initialisiert. Dieses Klassenattribut (auch als statisches Attribut bezeichnet) bezieht sich auf die gesamte Klasse, nicht auf ein Objekt einer Klasse. Die Definition einer statischen Variablen ermöglicht es uns, zu bestimmen, ob und wenn ja, welcher Turm ausgewählt wurde, indem wir auf den Klassennamen verweisen, ohne auf ein bestimmtes Objekt zugreifen zu müssen. Der Lehrer erklärt, dass es für das gesamte Spiel maximal eine manuell gesteuerte Instanz gibt. Zu Beginn sollte controlledInstance mit null initialisiert werden, da noch kein Tower zur manuellen Steuerung ausgewählt wurde.

Nach dem Überprüfen des internen Zustands der Objekte, ist der Unterschied zwischen statischen und nicht-statischen Attributen zu erklären. Der Lehrer bespricht mit den Schülerinnen und Schülern die Vorteile der Verwendung statischer Attribute. Der Lehrer sollte hier auch statische Methoden erwähnen und erklären, wo die Verwendung statischer Methoden von Vorteil ist

Commit*:* c4739460bed583d2126de066acc6b1149d022990

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/c4739460bed583d2126de066acc6b1149d022990

4. Methode der Klasse

Gegenstand: Grundlagen von Klassenmethoden

Zu besprechende Konzepte: Klassenmethoden, Klasse, Objekte

Aktivität: Es werden Klassenmethoden vorgestellt, die mit Daten auf Klassenebene arbeiten können. Die Notwendigkeit von Methoden wie changeControlledInstance wird erläutert, um das Umschalten des aktuell gesteuerten Turms zu verwalten. Diese Methoden können aufgerufen werden, ohne dass eine Instanz der Klasse erforderlich ist. Zum Beispiel läutet die Schulglocke für alle gleichzeitig, es spielt keine Rolle, welcher Schüler angesprochen wird, auf der anderen Seite erfordert die Überprüfung der Hausaufgaben eines Schülers Informationen über diesen bestimmten Schüler.

5. Aufgabe Task 7.7.: Auswahl des manuell gesteuerten Turms von einer zentralen Stelle

Gegenstand: Praktische Anwendung von Klassenmethoden

Zu besprechende Konzepte: Methoden, Klassenmethoden, Klassenattribute

Aktivität: Der Lehrer fügt die Methode changeControlledInstance() hinzu, um den manuell gesteuerten Turm auszuwählen. Der Parameter der Methode ist der Turm, den der Benutzer auswählen möchte. Zunächst sollte überprüft werden, ob die manuell gesteuerte Instanz bereits aktuell ausgewählt ist. Wenn dies der Fall ist, sollte sich nichts ändern. Anderenfalls sollte die aktuell manuell gesteuerte Instanz geändert werden, indem die Referenz auf die aktuell kontrollierte Instanz gesetzt wird. Testen Sie die Funktion manuell und beachten Sie, dass sich die Symbole der Türme nicht ändern. Es ist zu beachten, dass nur das Ändern der Referenz der kontrollierten Instanz die Steuerung nicht ändern würde und dass dies explizit ausgelöst werden muss. Fügen Sie den Code hinzu, der die vorher gesteuerte Instanz von der Steuerung löst, und fügen Sie nach dem Aktualisieren des Verweises Code hinzu, der die manuelle Steuerung der neu gesteuerten Instanz zuordnet. Es ist notwendig, Nullreferenzen zu überprüfen, die auftreten können, wenn derzeit keine kontrollierte Instanz vorhanden ist und wenn es keine neue manuell gesteuerte Instanz gibt (wenn der Parameter null ist).

Commit*:* 9dc6d8dd4dcbbd71edb8009c1a72403dea1a0ee0

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/9dc6d8dd4dcbbd71edb8009c1a72403dea1a0ee0

6. Aufgabe Task 7.8.: Wechsel des manuell gesteuerten Turms aufrufen

Gegenstand: Praktische Anwendung von statischen Klassenmethoden

Zu besprechende Konzepte: Methoden, statische Methoden, statische Attribute

Aktivität: Die Funktion changeControlledInstance() ist manuell zu testen, ob sie korrekt funktioniert. Anschließend ist mit den Schülern zu erarbeiten, wo diese Funktion aufgerufen werden soll. Die Methode sollte innerhalb der act()-Methode von Arena und innerhalb der processUserControl()-Funktion aufgerufen werden. Deklarieren Sie abschließend die Methode ManualTower.changeControl(Boolean) als privat, und beobachten Sie die Änderungen an der Instanz von ManualTower.

Commit*:* c052bbb6aa4c7e690d4d8cf55d3831028fa2b9e3

https://gitlab.kicon.fri.uniza.sk/oop4fun/project-tower-defense/commit/c052bbb6aa4c7e690d4d8cf55d3831028fa2b9e3

7. Wiederholung

Bei der Zusammenfassung sollte die Bedeutung von statischen Klassenattributen und statischen Methoden für die Verwaltung von gemeinsamen Eigenschaften aller Objekte eines Klassentyps hervorgehoben werden. Die Schüler sollen ermutigt werden, diese Konzepte in ihren eigenen Programmierprojekten anzuwenden.