

1 Teilnehmer/innen des Teams:

Klasse: BI19a	Team: Stählin Matteo Habtom Abrham
------------------	--

2 Anforderungsdefinition (Meilenstein A)

„Tower Defense“

Auftrag:
(Allgemeine Beschreibung)

Nutzen: Unterhaltung

Szenario:

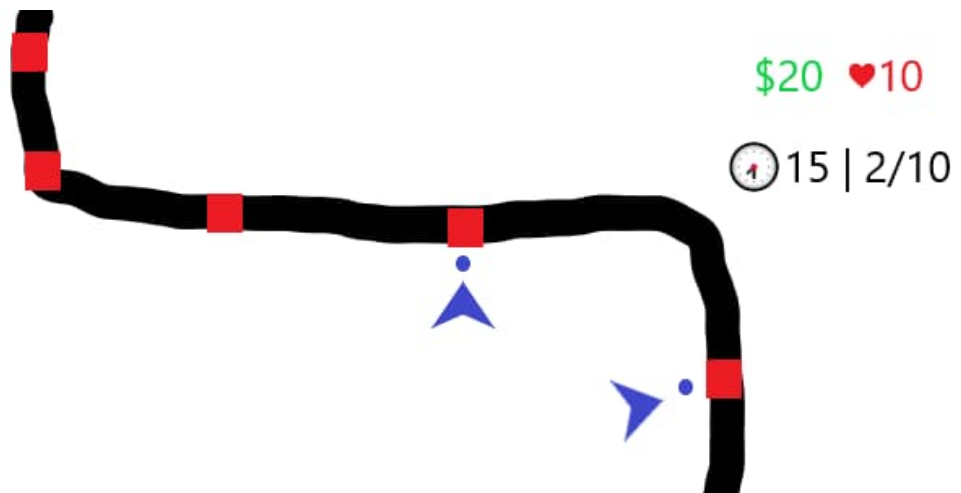
Tower Defense Game. Es gibt einen einfarbigen Pfad. Denn die Gegner folgen. Man soll auf Kästchen drücken können, um Türme zu platzieren die automatisch auf Gegner schießen. Man gewinnt, wenn man keine Gegner durchlässt. Es gibt einen Wave timer der die Zeit zwischen den Gegner spawns anzeigt und eine Anzeige auf welcher Wave man ist. Für besiegte Gegner erhält man Geld

Details:

- Pfad den Gegner folgen.
- Geld mit dem man Towers kaufen kann.
- Tower die auf Kugeln schießen.
- Leben die man verliert, wenn Gegner durchkommen.

Machbarkeitsabklärung:

- Gegner folgen der Farbe des Pfades.
-



MUSS Kriterien: (Konkrete Features, die umzusetzen sind)	Folgende Features sollen implementiert werden (Funktionalität): <ul style="list-style-type: none">• Geld mit dem man Towers kaufen kann.• Gegner die einem Pfad folgen.• Mehrere Levels.• Leben, die Man verliert falls ein Gegner den Pfad beendet.• Towers die auf Gegner schiessen.
KANN Kriterien: (Konkrete Features, die optional sind)	Folgende Features können zusätzlich implementiert werden: (Kreativität) <ul style="list-style-type: none">• Mehrere Gegnerarten (Schnellere oder solche mit mehr Leben).• Mehrere Tower arten.• Magie mit Abklingzeit die man irgendwo einsetzen kann.

2.1 Planung LB2

<i>MS</i>	<i>Tätigkeit / Abgabe</i>	<i>Soll-Datum</i>	<i>Ist-Datum</i>
A	Projektstart <ul style="list-style-type: none">➤ Team Bildung➤ Wahl / Ausarbeitung der Anforderungsdefinition Abnahme Anforderungsdefinition durch Lehrperson		
B	Teamaufgabe 1: <ul style="list-style-type: none">➤ Abgabe: Lösungsdesign (Analyse, Design: Funktionsmodell, UseCase, GUI, Storyboard)		
B2	Teamaufgabe 2: <ul style="list-style-type: none">➤ Abgabe: Testvorschrift und Testfälle		
C	Einzelaufgabe 3: <ul style="list-style-type: none">➤ Abgabe Szenario (.zip) mit Inline-Dokumentation, Systemdokumentation (UML Klassen-, Sequenzdiagramm)➤ Fachgespräch Projektabnahme		
C2	Einzelaufgabe 4: <ul style="list-style-type: none">➤ Abgabe: Ausgefüllter Systemtest		

3 Lösungsdesign (Meilenstein B: Teamaufgabe 1)

Anhand der Analyse wurde folgendes Lösungsdesign entworfen:

3.1 Funktionsmodell

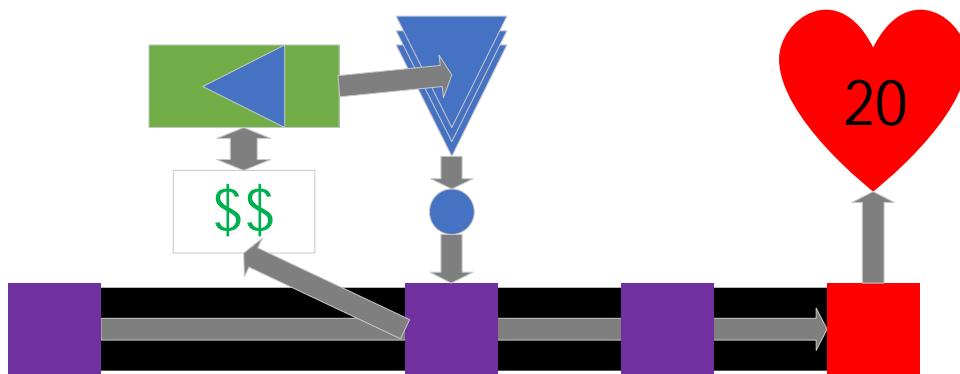
Im Folgenden sind die erwarteten Eingaben und Ausgaben beschrieben / dargestellt:

Objekte:

Tower, Bullet, Enemy, Goal, Button_Buy, Geld

Konzepte:

Gegnerauswahl, Pfadfolgen, Schiessen, Goal erreicht, Tower kaufen,

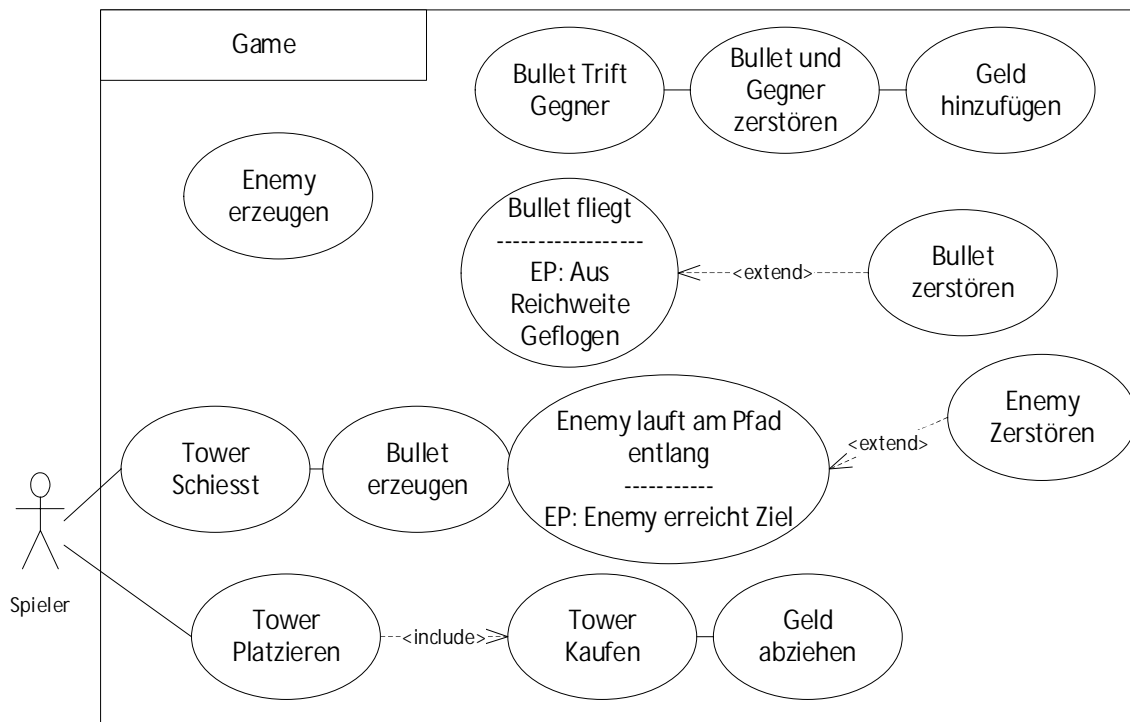


Legende:

- Enemy folgt dem Pfad zum Goal
- Button_Buy kauft Tower mit Geld und platziert ihn
- Tower sieht Enemy innerhalb Reichweite
- Tower schiesst auf Gegner (Manuel oder Automatisch)
- Bullet fliegt zum Enemy
- Bullet zerstört Enemy
- Zerstörter Enemy erzeugt Geld
- Enemy der zum Goal kommt nimmt leben weg

3.2 Anwendungsfälle (UseCases)

Folgende Anwendungsfälle sind hier detailliert dokumentiert:



Legende:

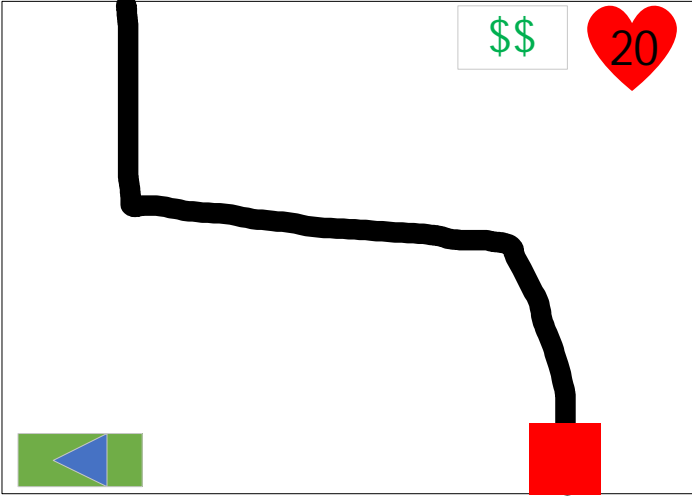
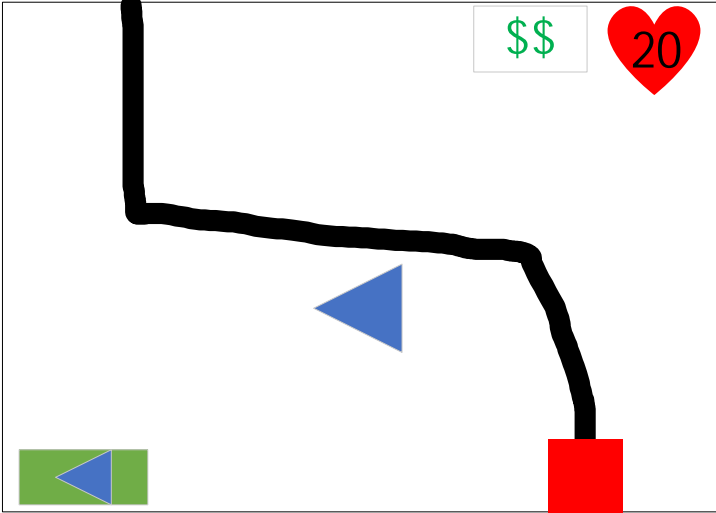
- Sobald der Turm (Tower) gekauft und platziert ist, wird das Geld abgezogene
- Enemy (Gegner) wird automatisch erzeugt
- Enemy (Gegner) läuft am Pfad und wird zerstört, sobald er das Ziel erreicht hat.
- Sobald das Bullet ausser Reichweite geflogen ist, wird er zerstört
- Der Turm (Tower) schiesst und Kugel (Bullet) wird erzeugt
- Die Kugel (Bullet) trifft Gegner (Enemy) beide wird zerstört und Geld wird hineingelegt

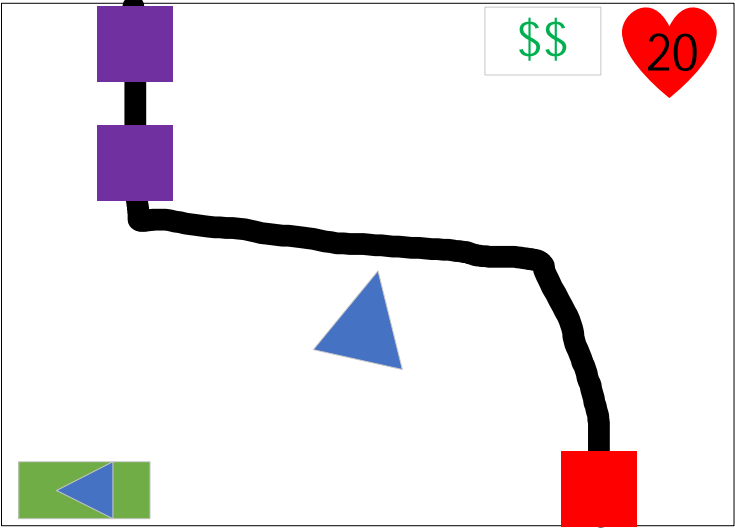
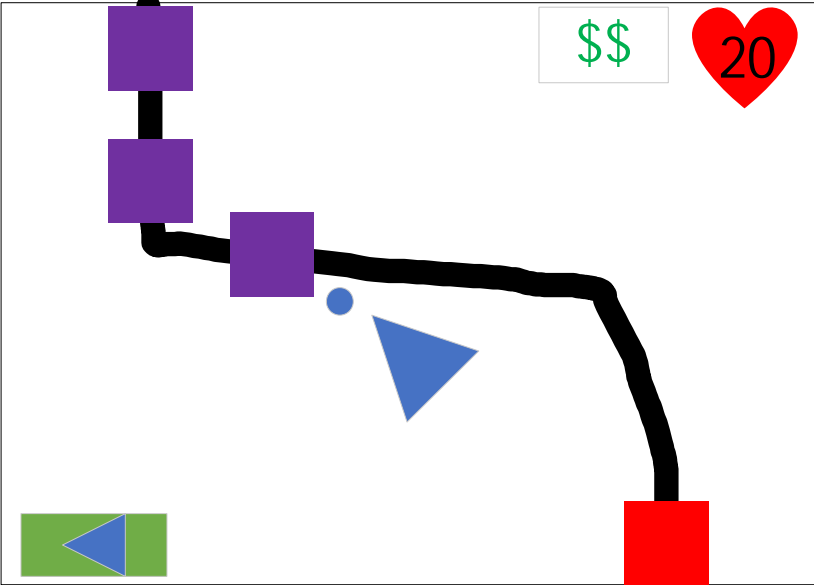
3.3 Ablauf

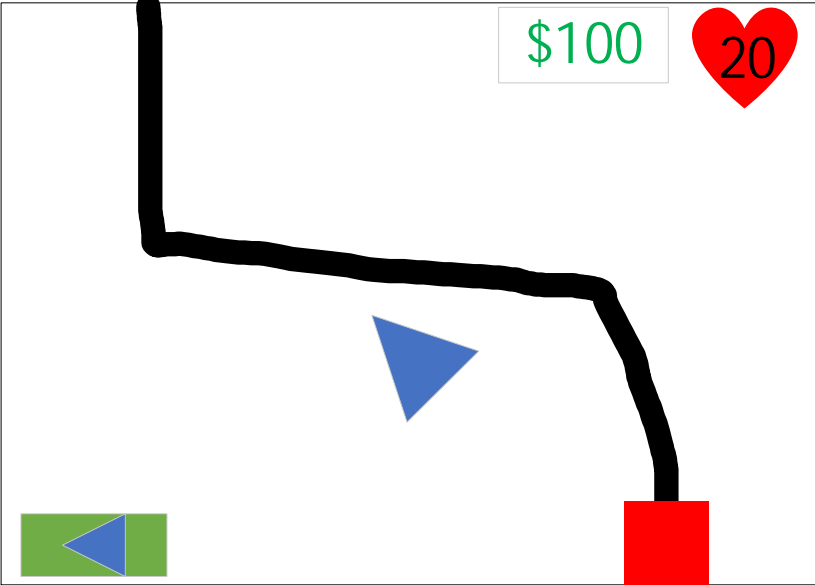
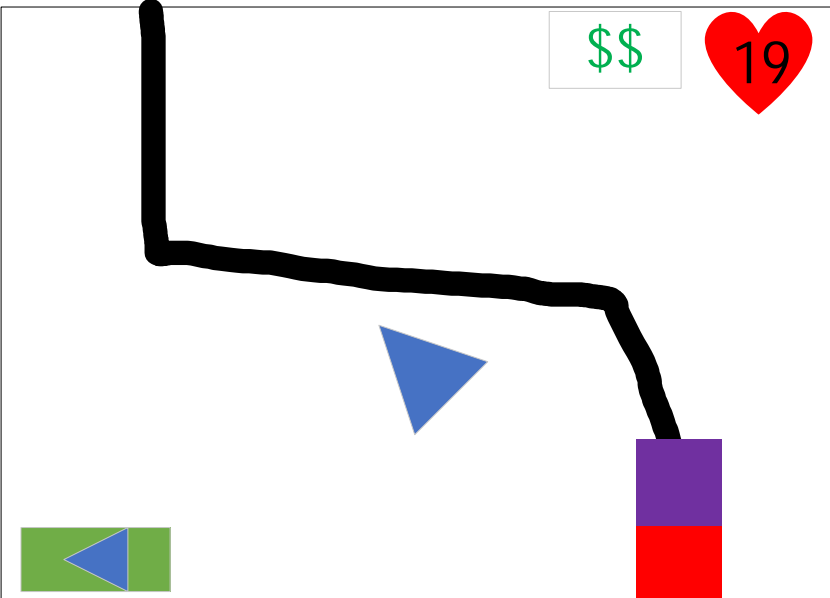
Aus Benutzersicht ist folgender Ablauf des Programms zu erwarten:

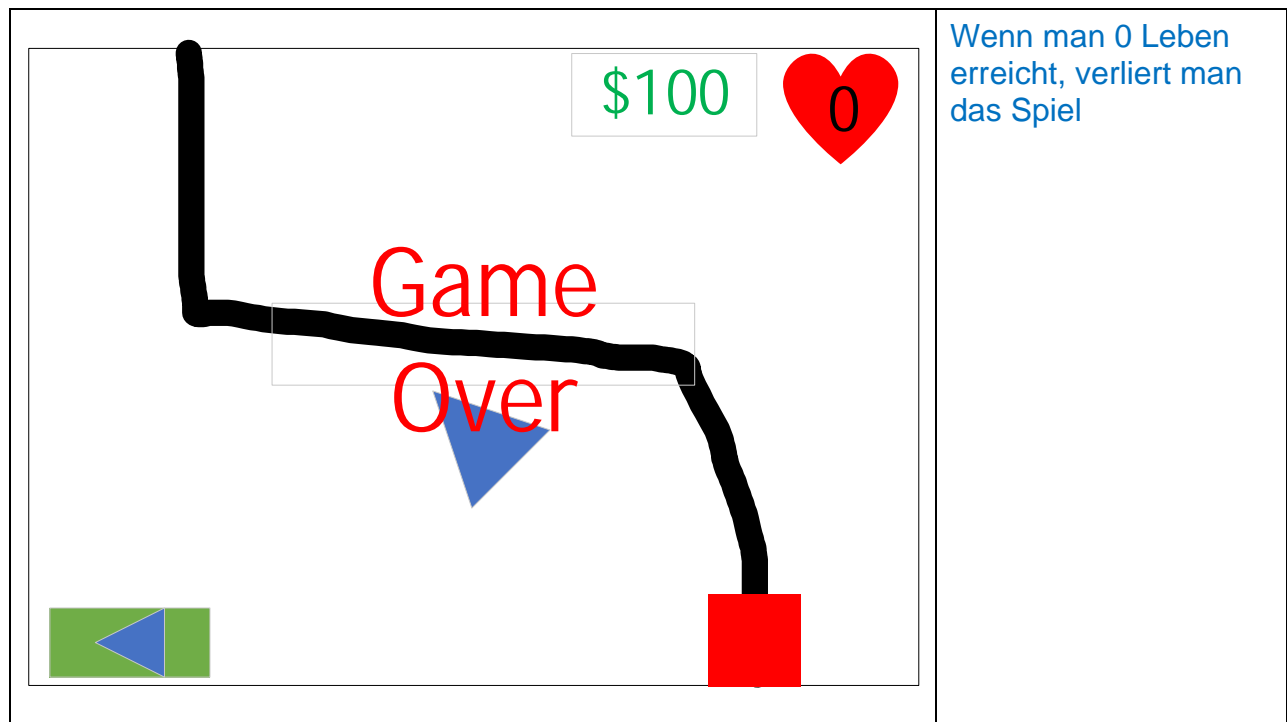
...

(Storyboard)

	<p>Startsituation</p> <p>Szenario muss gestartet werden Run></p> <p>Die Mappe wird eingerichtet mit Goal, Buy Buttons, Geld und Leben.</p>
	<p>Useraktivität</p> <p>Tower kann mit genügend Geld gekauft werden.</p>

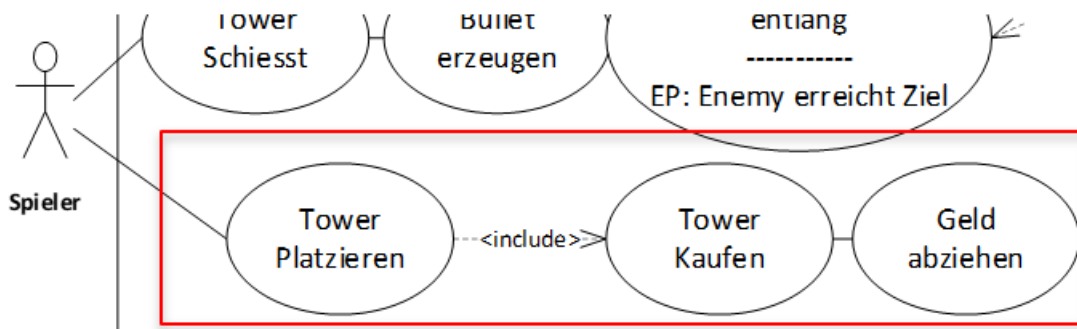
	<p>Enemys spawnen nach einer Zeit und Folgen dem Pfad zum Ziel.</p>
	<p>Useraktivität Entweder Tower oder Spieler schiesst an die Enemy zu.</p>

	<p>Getötete Gegner geben Geld.</p>
	<p>Ein Gegner, der das Ziel erreicht zieht leben ab.</p>



4 Testvorschrift (LB2 Meilenstein B2: Teamaufgabe 2)

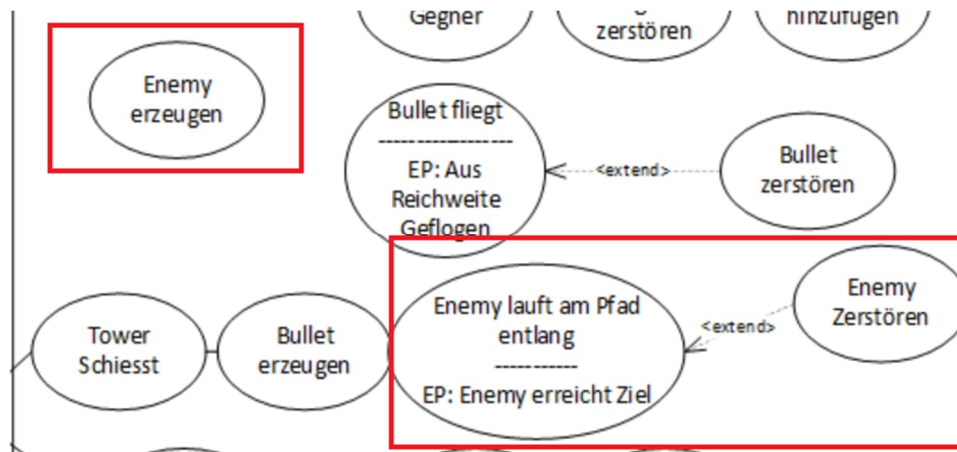
Projektname	Tower Defense
Version (getestetes Programm)	
Projekt-Code (Dateien)	
Fachlicher Ansprechpartner (Namen der Lehrperson)	
Autor des Testprotokolls	
Testdatum	
Name Tester	



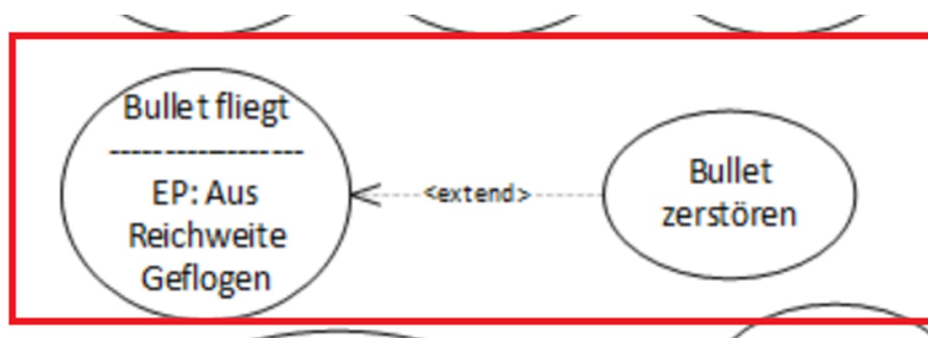
Use-Case		Testfall			
UC „Tower Kaufen“:		Test-Case „Tower Kaufen“: Der Tower wird gekauft nach dem er platziert wurden. Danach wird das Geld abgezogen.			
Akteure: Spieler Precondition: Spieler platziert Tower. Ereignis: Tower wird platziert und Geld wird abgezogen.		Trace 01: Keine Fehlerbehandlung oder Ausnahmesituation.			
#	Ablauf UC	Testaktivität (Input)	Erw. Resultat System/Benutzer	Tatsächliches Resultat	OK
1	Buy Botton wird geklickt.	TempTower wird erstellt wenn genügend Geld vorhanden.	Nach dem Klicken auf dem Buybutton folgt ein Tower der Maus.		
2	Der TempTower wird bewegt.	TempTower setzt seine Position gleich der Maus.	Tower folgt der Maus.		
3	Der TempTower wird platziert.	TempTower wird angeklickt und somit platziert er einen Tower Falls die Maus nicht über dem Pfad ist.	Nach dem Klicken wird fangt der Tower an zu schiessen.		
4	Das Geld wird abgezogen.	Das Geld wird abgezogen und der TempTower gelöscht. Die Geldanzeige wird angepasst.	Die Geldanzeige wird reduziert.		
Postcondition: Tower schiesst.		Postcondition:			

Use-Case		Testfall			
UC „Tower Kaufen“:		Test-Case „Tower Kaufen“:			
Akteure: Spieler Precondition: Spieler platziert Tower. Ereignis: Butten wird gedrückt aber nichts passiert		Trace 02: Der Spieler hat nicht genügend Geld.			
#	Ablauf UC	Testaktivität (Input)	Erw. Resultat System/Benutzer	Tatsächliches Resultat	OK

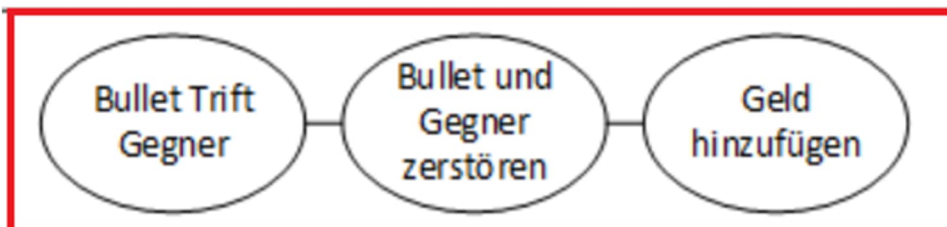
1	Buy Button wird geklickt.	Spieler hat nicht genügend Geld deshalb wird kein TempTower erstellt.	Sieht Fehlermeldung das nicht genügend Geld vorhanden ist.		
Postcondition: Fehlermeldung wird entfernt.		Postcondition:			



Use-Case		Testfall			
UC „Enemy läuft am Pfad entlang“:		Test-Case “Enemy läuft am Pfad entlang“:			
Akteure: Enemy Precondition: Ereignis: Ein Gegner läuft dem Pfad entlang		Trace 01: Keine Fehlerbehandlung oder Ausnahmesituation.			
#	Ablauf UC	Testaktivität (Input)	Erw. Resultat System/Benutzer	Tatsächliches Resultat	OK
1	Gegner wird erstellt	Ein Gegner wird erzeugt nach den entsprechenden Timer.	Ein Gegner taucht auf.		
2	Gegner läuft dem Pfad entlang	Gegner bewegt sich am Pfad entlang und dreht sich, wenn er nicht mehr auf dem Pfad ist.	Der Gegner folgt dem Pfad.		
3	Gegner erreicht das Ziel	Der Gegner zieht Leben ab und zerstört sich danach.	Der Gegner verschwindet.		
Postcondition: Getestet ob Leben noch über 0 sind.		Postcondition: Getestet ob Leben noch über 0 sind.			



Use-Case		Testfall			
UC „Bullet fliegt“:		Test-Case “Bullet fliegt“:			
Akteure: Bullet Precondition: Tower schiesst Ereignis: Bullet fliegt und zerstört sich nach dem sie ausser Reichweite ist.		Trace 01: Keine Fehlerbehandlung oder Ausnahmesituation.			
#	Ablauf UC	Testaktivität (Input)	Erw. Resultat System/Benutzer	Tatsächliches Resultat	OK
1	Bullet fliegt in die geschossene Richtung.	Bullet bewegt sich in die entsprechende Richtung.	Bullet fliegt.		
2	Bullet überwindet die Range des Towers und zerstört sich selber.	Bullet schaut, ob sie schon ausser Reichweite geflogen ist.	Bullet fliegt und verschwindet.		
Postcondition:		Postcondition:			



Use-Case		Testfall			
UC „Bullet fliegt “:		Test-Case “Bullet fliegt “:			
Akteure: Bullet Precondition: Tower schiesst Ereignis: Bullet fliegt und zerstört sich und Gegner.		Trace 01: Keine Fehlerbehandlung oder Ausnahmesituation.			
#	Ablauf UC	Testaktivität (Input)	Erw. Resultat System/Benutzer	Tatsächliches Resultat	OK
1	Bullet fliegt in die geschossene Richtung.	Bullet bewegt sich in die entsprechende Richtung.	Bullet fliegt.		

2	Die Bullet trifft ein Gegner.	Die Bullet zerstört sich selbst und Gegner. Es wird Geld hinzugefügt.	Die Bullet verschwindet nach dem der Gegner auch es wird Geld hinzugefügt.		
Postcondition:		Postcondition:			

Use-Case		Testfall			
UC „Bullet fliegt“:		Test-Case “Bullet fliegt“:			
Akteure: Bullet Precondition: Tower schießt Ereignis: Bullet fliegt und zerstört sich und Gegner.		Trace 01: Gegner hat noch zu viel Leben.			
#	Ablauf UC	#	Ablauf UC	#	Ablauf UC
1	Bullet fliegt in die geschossene Richtung.	Bullet bewegt sich in die entsprechende Richtung.	Bullet fliegt.		
2	Bullet überwindet die Range des Towers und zerstört sich selbst.	Bullet schaut, zieht am Gegner Leben ab. Gegner hat noch Leben.	Bullet fliegt und verschwindet.		
Postcondition:		Postcondition:			

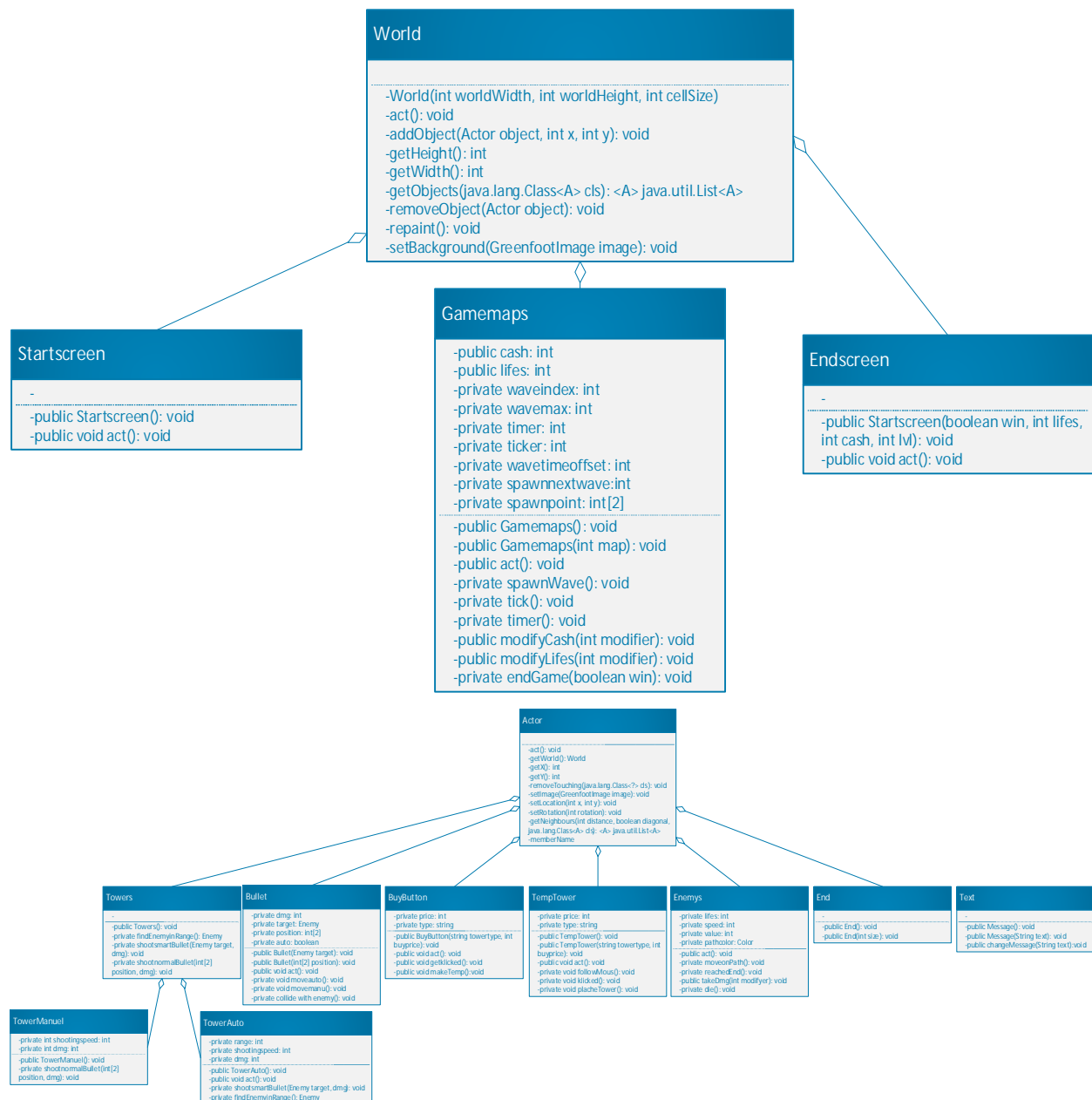
5 Systemdokumentation (Meilenstein C: individuelle Aufgabe 3)

Das erstellte Java-Projekt (Greenfoor-Szenario) ist hier detailliert abgelegt:

[M226B_Aufgabe_3_Szenario_IhrName.zip](#)

5.1 Statisches Design: Klassendiagramm

Folgend die statische Struktur des Szenarios



5.2 Umfang / Abgrenzung / Änderungen gegenüber Design

Aufgrund unten beschriebener Umstände sind Anpassungen des ursprünglichen Lösungsdesigns gemacht worden:

...

(Umstände / Anpassungen / Veränderungen)

5.3 Funktionalität der Implementation.

Zusätzlich zu der Inline-Dokumentation sind hier folgende Funktionen detailliert beschrieben:

...

(Ausführliche Beschreibung der internen Funktionen
oder Verweis zum Inline-Kommentar mit JavaDoc! (`/** @param @return **/`))

5.4 Dynamische Struktur: Sequenzdiagramm

Ein zentraler Ablauf eines UseCases ist im Folgenden dargestellt:

...

(Darstellung eines zentralen Ablaufs mittels Sequenzdiagramm)

Trace: ...

...

6 Bedienungsanleitung (**Meilenstein C: individuelle Aufgabe 3**)

...

7 Testprotokoll (**LB2 Meilenstein C2: individuelle Aufgabe 4**)

Ausgefülltes Testprotokoll siehe Dokument
[**M226B_LB2_Testvorschrift_MS-C2_Name.docx**](#)