Projekt zu GdI 1 - Wintersemester 2014/15

Guido Rößling, Sebastian Fach, Peter Klöckner

29. Januar 2015

Version 1.0

Inhaltsverzeichnis

1 Einführung in das Projekt **2**
1.1 Das Spiel *Gorillas* 2

2 Organisation des Projekts **3**

3 Ihre Aufgabe **5**
3.1 Ablauf des Code-Review 5
3.2 Dokumentation 6
3.3 Hinweise 6
3.4 Minimale Ausbaustufe 7
3.5 Ausbaustufe I 10
3.6 Ausbaustufe II 11
3.7 Ausbaustufe III 12
3.8 Bonuspunkte 13

4 Materialien **14**
4.1 Klasse *de.tu_darmstadt.gdi1.gorillas.main.Gorillas* 14
4.2 Die ersten Schritte 14
4.3 Basisereignisse und Basisaktionen 14

5 Zeitplanung **14**

6 Anpassungen für Studierende im Studiengang CE **15**

7 Liste der Anforderungen **17**

1 Einführung in das Projekt

Im Rahmen des Projekts implementieren die Student_innen[1] in Gruppen von jeweils vier Personen eine Java-Version des Spiels *Gorillas*. Die Aufgabenstellung stellt zunächst das zugrundeliegende Spiel vor.

Die Aufgabe kann in vier verschiedenen „Ausbaustufen“ bearbeitet werden, die jeweils eine unterschiedliche Punktzahl zur Gesamtnote beitragen. Die minimale Ausbaustufe muss zum Erreichen der Mindestpunktzahl **vollständig** implementiert werden.

Ab Ausbaustufe I können nicht erreichte Punkte der gegebenen Ausbaustufe durch Elemente höherer Ausbaustufen „ausgeglichen“ werden. Projektabgaben, die „zwischen“ Ausbaustufen liegen, bei denen also erwartete Inhalte fehlen oder zusätzliche Elemente eingebaut wurden, sind natürlich ebenfalls möglich; die Ausbaustufen geben nur eine grobe Orientierung vor. Beachten Sie dabei jedoch, dass die Ausbaustufen nach Schwierigkeitsgrad gruppiert sind, d.h. Aufgaben höherer Stufen sind in der Regel schwieriger zu lösen als Aufgabenteile niedrigerer Ausbaustufen.

1.1 Das Spiel *Gorillas*

Im Spiel *Gorillas* spielen Sie Eins gegen Eins. Jede Person steuert dabei einen Gorilla und versucht den gegnerischen Gorilla mit einer Banane abzuwerfen. Die Gorillas stehen in gewisser Entfernung, einer links und einer rechts, auf verschiedenen Hochhäusern einer Skyline. Das Spiel ist rundenbasiert: Zunächst tätigt der/die erste Spieler_in den Wurf. Trifft dieser Wurf, so wurde die Runde gewonnen, ansonsten ist der/die nächste Spieler_in an der Reihe. Ein Wurf wird durch zwei Eingaben gesteuert: den Abwurfwinkel und die Wurfgeschwindigkeit. Der Winkel wird als Gradzahl von 0 bis 360 angegeben, die Geschwindigkeit auf einer Skala von 0 bis 200.

Abbildung 1 auf der nächsten Seite zeigt ein mögliches Menü. Abbildung 2 auf Seite 4 zeigt ein mögliches Spielfenster. Wir erwarten nicht, dass das im Rahmen des Projekts erstellte Spiel der Ausgabe optisch ähnlich sieht; die Abbildung soll nur zum besseren Nachvollziehen dienen.

Auf einem beliebig großen rechteckigen Spielfeld kann es folgende Spielobjekte geben:

Gorilla Jeder Spieler verfügt über einen Gorilla. Folglich befinden sich zum Start zwei Gorillas, einer links und einer rechts, auf dem Feld. Gorillas unterschiedlicher Parteien müssen nicht visuell unterscheidbar sein.

Hochhaus Die Gorillas stehen auf Hochhäusern, die eine Skyline bilden.

Banane Tätigt ein Spieler einen Wurf, so fliegt eine Banane (hoffentlich) in Richtung des gegnerischen Gorillas.

[1] Diese Schreibweise wird durchgängig verwendet, um alle möglichen Personen gleichberechtigt anzusprechen, auch wenn sie für den_die Leser_in vielleicht teilweise irritierend wirkt. Betrachten Sie dies als Reflektionsmöglichkeit! Mehr Informationen unter `http://de.wikipedia.org/wiki/Geschlechtergerechte_Sprache#Sichtbarmachung` bei „Gender Gap“.

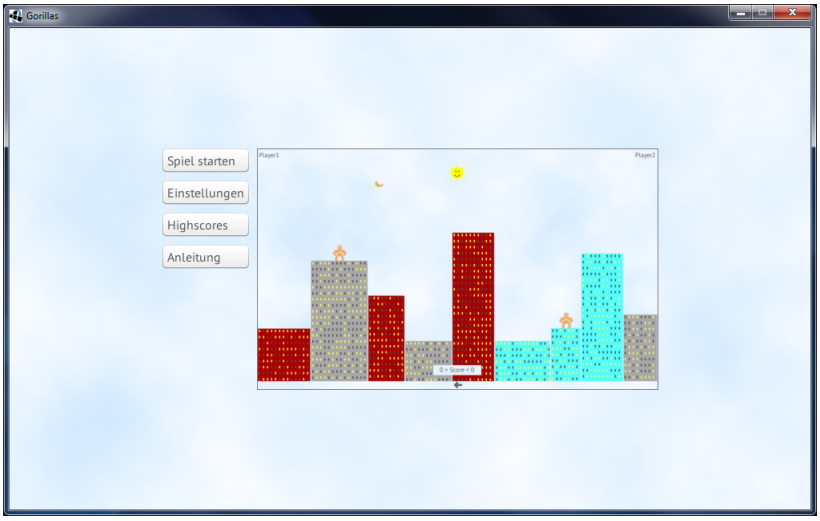

Abbildung 1: Beispiel einer grafischen Umsetzung des Menüs von Gorillas

2 Organisation des Projekts

Der offizielle Bearbeitungszeitraum des Projekts beginnt am *Mittwoch, den 04.03.2015* und endet am *Freitag, den 20.03.2015*. Zur Teilnahme am Projekt müssen sich *alle* Mitglieder einer gegebenen Projektgruppe im Portal im extra für das Projekt angelegten Kurs angemeldet haben und der gleichen Projektgruppe beigetreten sein. Wir raten **dringend** dazu, sich möglichst früh in einer gemeinsamen Projektgruppe anzumelden. Bitte beachten Sie, dass bei der Eintragung in die Projektgruppe nur Studierende einer Projektgruppe beitreten können, die bereits im Kurs zum Projekt *angemeldet* sind *und noch keiner Projektgruppe beigetreten sind*. Gruppen, die am Ende des Anmeldeintervalls noch nicht die Größe 4 haben, werden von uns mit zufällig gewählten anderen Studierenden aufgefüllt. Bitte versuchen Sie dies nach Möglichkeit zu vermeiden!

Die Aufgabenstellung wird etwa vier Wochen vor Beginn der Projektphase, d.h. am *Mittwoch, den 04.02.2015*, im Portal veröffentlicht. Ab diesem Zeitpunkt ist prinzipiell bereits die Bearbeitung der Aufgabe möglich. Da zu diesem Zeitpunkt die Gruppen im Lernportal noch nicht endgültig gebildet werden konnten, raten wir in diesem Fall aber *dringend* zu einer Anmeldung als Gruppe von vier Student_innen in der entsprechenden Anmeldung im Lernportal.

Im Projekt bearbeitet die Gruppe die in den folgenden Abschnitten näher definierte Aufgabe. Dabei wird die Gruppe von den Veranstaltern wie folgt unterstützt:

- Das Portal und insbesondere der neu angelegte Kurs zum *Projekt im Wintersemester 2014/15* kann für alle gruppenübergreifenden Fragen zum Verständnis der Aufgabe oder Unklarheiten bei der Nutzung der Vorlagen genutzt werden.
- Jede Projektgruppe erhält im Portal eine eigene *Gruppe* sowie ein *Projektgruppenforum*. In diesem Projektgruppenforum können Fragen diskutiert oder Code-Fragmente ausgetauscht werden (Tipp: umgeben Sie Code im Portal immer mit `[code java] ... [/code]`, damit er besser lesbar ist). Da die jeweiligen Tutor_innen der Gruppe ebenfalls der Projektgrup-

Abbildung 2: Beispiel einer grafischen Umsetzung des Spielfensters von Gorillas

pe angehören werden, sind sie in die Diskussionen eingebunden und können leichter und schneller Feedback geben.

Bitte beachten Sie, dass diese Projektgruppen im Portal von uns nur ein *Angebot* an Sie sind, das Sie nicht nutzen müssen. Wenn Sie beispielsweise alle Aufgaben gemeinsam in einer WG erledigen, bringt eine Abstimmung über das im Portal erstellte Projektgruppenforum vermutlich mehr Aufwand als Nutzen.

- Eine Gruppe von Tutor_innen betreut das Projekt. Allen Tutor_innen wird dabei eine gewisse Anzahl Projektgruppen zugeteilt. Die Aufgabe der Tutor_innen ist es, die Gruppe im Rahmen des Projekts bei offenen Fragen zu unterstützen, nicht aber bei der tatsächlichen Implementierung. Insbesondere helfen die Tutor_innen nicht bei der Fehlersuche und geben auch keine Lösungsvorschläge. Die Tutor_innen stehen der Gruppe auch nur zeitlich begrenzt zur Verfügung: pro Gruppe wurden bis zu drei Stunden Betreuung sowie eine halbe Stunde für die Testierung angesetzt.

- Für den einfacheren Einstieg stellen wir einige Materialien bereit, die Sie im Portal herunterladen können. Diese Materialien inklusive vorgefertigten Klassen *können* Sie nutzen, müssen es aber nicht. Zu den Materialien zählen auch vorbereitete Testfälle. *Diese **müssen** unverändert auf Ihrer Implementierung funktionieren, da sie—zusammen mit „privaten" Tutorentests—als Basis für die Abnahme dienen.* Dabei darf auch der Pfad zu den Testfällen *nicht* verändert werden.

Wir weisen *ausdrücklich* darauf hin, dass Sie sich **möglichst früh** mit den Tests vertraut machen sollten, um unliebsame Überraschungen „kurz vor Fertigstellung" zu vermeiden!

Die *Abnahme* oder *Testierung* des Projekts (beschrieben in Abschnitt 3) erfolgt durch den_die Tutor_in der Gruppe und erfordert eine *vorherige Terminabsprache*. Bitte bedenken Sie, dass auch unsere Tutor_innen Termine haben (etwa die Abnahme der anderen von ihnen betreuten Gruppen)

und nicht „pauschal immer können“. In Ihrem eigenen Interesse sollten Sie daher versuchen, so früh wie möglich einen Termin für die Besprechungen und die Abnahme zu vereinbaren.

Sie sollten auch einen Termin für die erste Besprechung mit dem_r Tutor_in absprechen. Vor diesem Termin sollten Sie schon dieses Dokument komplett durchgearbeitet haben, sich alle offenen Fragen notiert haben (und im Portal nach Antworten gesucht haben), *und* einen Entwurf vorbereitet haben, wie die Lösung Ihrer Gruppe aussehen soll. Dieser Entwurf kann in UML erfolgen, aber prinzipiell ist jede (für den_die Tutor_in) lesbare Form denkbar. Bitte bringen Sie diesen Entwurf zum ersten Treffen mit dem_der Tutor_in mit, damit Sie direkt Feedback erhalten können, ob dieser Ansatz funktionieren kann. Auch hier kann die Nutzung des Portals mit dem Projektgruppenforum helfen, den_die Tutor_in „früher zu erreichen“.

3 Ihre Aufgabe

Implementieren Sie eine lauffähige Java-Version des Spiels *Gorillas*, die mindestens der „minimalen Ausbaustufe“ entspricht.

Das fertige Spiel muss von dem_der Tutor_in *vor Ende des Projekts testiert werden*. Dazu müssen die Dokumentation (etwa 2 DIN A4-Seiten) sowie der Source-Code und alle zum Übersetzen notwendigen Bibliotheken und Dateien—außer den von uns im Portal bereitgestellten—rechtzeitig vor Ablauf der Einreichung **von *einem* Gruppenmitglied im Portal hochgeladen werden**.

Das Testat besteht aus den folgenden drei Bestandteilen:

Live-Test Der_die Tutor_in startet das Spiel und testet, ob alles so funktioniert wie spezifiziert. Dazu werden potenzielle bestimmte vorgegebene Szenarien durchgespielt, aber auch zufällig „herumgespielt“.

Software-Test Der_die Tutor_in testet die Implementierung mit den für alle Teilnehmer_innen bereitgestellten (öffentlichen) und nur für die Tutor_innen und Mitarbeiter_innen verfügbaren (privaten) JUnit-Tests. Alle Tests müssen **ohne Benutzerinteraktion** abgeschlossen werden können.

Code-Review Der_die Tutor_in sieht sich den Quellcode sowie die Dokumentation an und stellt Fragen dazu.

3.1 Ablauf des Code-Review

Im Hinblick auf den Code-Review sollten Sie auf gut verständlichen und dokumentierten Code sowie eine sinnvolle Klassenhierarchie achten, in der Regel auch mit Aufteilung der Klassen in Packages.

Der_die Tutor_in wird einzelne Gruppenmitglieder seiner Wahl zu Teilen des Quelltexts befragen. Daher sollte sich jedes Gruppenmitglied mit allen Codeteilen auskennen—der_die Tutor_in wählt aus, zu *welchem Thema* eine Frage gestellt wird und wählt auch aus, *wer* die Frage beantworten soll. Die Bewertung dieses Teils bezieht sich also auf die Aussage eines „zufällig ausgewählten“ Gruppenmitglieds, geht aber in die Gesamtpunktzahl der Gruppe ein.

Damit soll einerseits die „Trittbrettfahrerei“ reduziert werden („ich habe zwar nichts getan, will aber dennoch die Punkte haben“). Gleichzeitig fördert diese Regelung die Gruppenarbeit, da auch

und gerade besonders „starke“ Mitglieder verstärkt Rücksicht auf „schwächere“ nehmen müssen—sonst riskieren sie eine schlechtere Punktzahl, wenn „der_die Falsche“ gefragt wird. Durch eine entsprechend bessere Abstimmung in der Gruppe steigen die Lernmöglichkeiten **aller** Gruppenteilnehmer. Auch für (vermeintliche?) „Expert_innen“ wird durch das Nachdenken über die Frage „wie erkläre ich das verständlich?“ das eigene Verständnis vertieft.

3.2 Dokumentation

Neben dem Quelltexten ist auch eine kurze Dokumentation abzugeben (etwa 2 DIN A4-Seiten). Diese sollte die *Klassenstruktur* ihrer Lösung in UML umfassen und kurz auf die in ihrer Gruppe *aufgetretenen Probleme* eingehen sowie *Feedback zur Aufgabenstellung* liefern. Nur die Klassenstruktur geht in die Bewertung ein; die anderen Elemente helfen uns aber dabei, das Projekt in der Zukunft besser zu gestalten und sind daher für uns sehr wichtig. Sie können den Teil mit der (hoffentlich konstruktiven) Kritik am Projekt auch gerne separat auf Papier—auf Wunsch ohne Angabe des Gruppennamens—dem_der Tutor_in geben, wenn Sie das Feedback lieber anonym geben wollen.

Für die Erstellung der Klassendiagramme können Sie beispielsweise die folgenden Tools nutzen:

- *doxygen* (`http://www.stack.nl/~dimitri/doxygen/`) ist eine Alternative zu JavaDoc, die—bei Wahl der entsprechenden Optionen—auch Klassendiagramme erzeugt. Eine Dokumentation zu *doxygen* finden Sie auf der obenstehenden Projekt-Homepage.
- *Fujaba* (`http://wwwcs.uni-paderborn.de/cs/fujaba/`)
- *BlueJ* (`http://bluej.org/`)

Dies sind nur unsere Empfehlungen. Es steht Ihnen selbstverständlich frei, andere Tools zu nutzen, etwa OpenOffice Draw oder Microsoft Word. Bitte reichen Sie die Dokumentation und insbesondere das Klassendiagramm als **PDF-Datei** ein.

Zusätzlich sind mindestens *vier* repräsentative Screenshots Ihres Spiels einzureichen, darunter ein Bild vom Startmenü.

3.3 Hinweise

Denken Sie bitte daran, dass **Testfälle keine Interaktion mit den Spieler_innen erfordern dürfen**.

Sollten Sie sich für die Nutzung des vorgegebenen Frameworks entscheiden, so nutzen Sie das Konzept von Entitäten, Ereignissen und Aktionen. Die Tutor_innen werden bewerten, wie gut Ihnen das gelungen ist.

Sollten Sie nicht das vorgegebene Framework verwenden, so sollten Sie in Ihrem Code strikt zwischen Logik (Code) und Darstellung (Design) unterscheiden. Die Tutor_innen werden bewerten, wie gut diese Trennung bei Ihnen gelungen ist.

3.4 Minimale Ausbaustufe

Um das Projekt bestehen zu können, also mindestens 50 aus 100 Punkten zu erhalten, müssen **alle** nachfolgend genannten Leistungen erbracht werden.

Fehlende Punkte aus der minimalen Ausbaustufe können nur in Ausnahmefällen ausgeglichen werden.

Pünktliche Abnahme—0 Punkte Der Quelltext mit Dokumentation wird rechtzeitig in das Portal hochgeladen (als JAR oder ZIP mit allen für Übersetzung und Ausführung erforderlichen *.java* Dateien, Bildern etc.) und wird von dem_der Tutor_in nach vorheriger Terminabsprache rechtzeitig vor dem Ablaufen der Abnahmefrist testiert. **Die Projektgruppe** ist für die Einhaltung der Termine verantwortlich.

Compilierbares Java—0 Punkte Der Quelltext ist komplett in Java implementiert und kann separat ohne Fehlermeldung neu compiliert werden.

Nur eigener Code—0 Punkte Der Quelltext enthält **keine** oder **nur genehmigte** fremde Codeteile, insbesondere keinen Code von anderen Projektgruppen oder aus dem Internet. Die Nutzung von fremdem Code kann nur durch die Mitarbeiter_innen (nicht die Tutor_innen!), individuell oder bei allgemeiner Nachfrage im Portal pauschal für konkrete Codeteile oder Bibliotheken, genehmigt werden.

In Ihrem eigenen Interesse müssen Sie in der kurzen Ausarbeitung sowie beim Testat **explizit und unaufgefordert** auf fremde Codeteile (mit Angabe der Quelle und des Umfangs der Nutzung) hinweisen, um sich nicht dem Vorwurf des Plagiarismus oder gar des Betrugsversuchs auszusetzen. Wir behalten uns vor, Lösungen (auch automatisiert) miteinander zu vergleichen. Bitte beachten Sie, dass wir Ihre Abgabe ab einem gewissen Umfang der Nutzung von fremden Codes nur noch als gescheitert betrachten können, da der Eigenanteil zu gering wird.

Vorbereitung für automatisierte Tests—0 Punkte Um die öffentlichen Tests korrekt ablaufen zu lassen, müssen Sie die Klasse `GorillasTestAdapterMinimal` korrekt implementieren. Diese Klasse soll *keine* Spiel-Funktionalität enthalten, sondern die einzelnen Methoden lediglich auf die entsprechenden Methoden in Ihrer Implementierung abbilden.

Objekte die Sie in diesem Adapter benötigen können Sie im Konstruktor erzeugen und „passend“ initialisieren.

Beispiel: Haben Sie sich entschieden, die vergangene Zeit der aktuellen Karte in dem statischen Feld `elapsedTime` der Klasse `StopWatch` zu speichern, so wäre dies eine passende Implementierung der Methode `getElapsedTime()` der Klasse `GorillasTestAdapterMinimal`:

```
public boolean getElapsedTime() {
    return StopWatch.elapsedTime;
}
```

Haben Sie sich für einen anderen Entwurf entschieden, sähe die Implementierung anders aus.

Ihr Programm muss auch *ohne diese Klasse voll funktionsfähig sein!* In Eclipse können Sie das testen, indem Sie die Klasse vom Build ausschließen (Rechtsklick auf `GorillasTestAdapterMinimal`, Build path, exclude).

Öffentliche Tests sind erfolgreich—0 Punkte Die öffentlichen Testfälle der Klasse `GorillasTestsuiteMinimal` werden fehlerfrei absolviert. Da Sie diese bereits während des Projekts selbst testen können, sollte diese Anforderung für Sie kein Problem darstellen. Beachten Sie, dass Sie für den Ablauf der Tests die Klasse `GorillasTestAdapterMinimal` implementieren müssen!

Generell sind die öffentlichen Tests eine Hilfe für Sie. Die Testfälle durch „Schummeln" erfolgreich zu bestehen, bringt Sie nicht weiter.

Grafische Benutzerschnittstelle—3 Punkte Es gibt eine grafische Benutzerschnittstelle („GUI"), mit der man *Gorillas* spielen kann.

Spielmenü—2 Punkte Das Spiel Gorillas muss in einem Spielmenü beginnen. Das Menü muss mindestens jeweils einen Button „Neues Spiel starten" und „Beenden" mit entsprechender Semantik enthalten. Im Spiel soll das Spielmenü durch Drücken der **Escape**-Taste erscheinen und bei nochmaligem Drücken soll wieder das Spiel selbst angezeigt werden.

Hinweis: Sollten Sie nicht das *eea*-Framework nutzen, so dürfen Steuerungsbefehle im Menü keine Auswirkung auf Spielzüge haben. Während das Menü sichtbar ist, ist das Spiel komplett pausiert. Im *eea*-Framework ist dieses Verhalten bereits standardmäßig der Fall.

Neues Spiel starten—2 Punkte Der_die Spieler_in soll über die Taste **n** aus dem Hauptmenü oder Klicken auf einen „Neues Spiel"-Button heraus ein neues Spiel starten können. Drückt ein_e Spieler_in im Hauptmenü **n**, oder klickt auf einen „Neues Spiel"-Button, so lassen sich in einem neuen GameState, dem GameSetupState, die zwei Namen der Spieler_innen angeben.

Eingabe und Anzeige der Spielernamen—6 Punkte Im GameSetupState können die zwei Spieler_innen ihre gewünschten Namen eingeben. Es sollte nicht erlaubt sein, leere Namen oder zwei gleiche Namen einzugeben.

Über einen „Spiel starten"-Button lässt sich das Spiel nun tatsächlich beginnen. Sind beide Namen korrekt eingegeben, so wechselt das Spiel in den GamePlayState. Ist dies nicht der Fall, so bleibt das Spiel im GameSetupState. In diesem Fall wird zu jedem Namenseingabefeld, in das ein nicht zulässiger Name eingegeben wurde, eine entsprechende Fehlermeldung in der Nähe des Feldes angezeigt.

Die zwei eingegebenen Namen der Spieler_innen werden während des Spielens angezeigt, etwa links und rechts oben oder jeweils in der Nähe des Gorillas der jeweiligen Person.

Grafische Umsetzung der Objekte—4 Punkte Für jedes Spielobjekt und jeden Spielzustand existiert eine passende grafische Umsetzung gemäß der Spielbeschreibung.

Platzierung der Gorillas—3 Punkte Ein Gorilla wird zu Spielbeginn auf dem ersten bis dritten Hochhaus von links, der andere auf dem ersten bis dritten Hochhaus von rechts platziert.

Eingabe der Wurfparameter—6 Punkte Das Spiel soll mit der Tastatur und der Maus gesteuert werden können. Für den_die Spieler_in, der_die gerade an der Reihe ist, einen Wurf zu parametrisieren, werden im GamePlayState zwei Input-Felder und ein Wurf-Button angezeigt.

In eines der Eingabefelder lässt sich ein Winkel als natürliche Zahl im Bereich von 0 bis 360 eingeben. In das andere Eingabefeld lässt sich die Geschwindigkeit ebenfalls als natürliche Zahl, aber im Bereich von 0 bis 200 eingeben. Andere Eingaben sollen jeweils nicht möglich sein. Damit der_die Spieler_in weiß, was er_sie in die Input-Felder eingeben soll, sind diese Felder beschriftet. So steht beispielsweise links neben dem einen Eingabefeld "Winkel:" und links neben dem anderen Eingabefeld "Geschwindigkeit:".

Kann gerade kein Wurf parametrisiert werden, d. h. wenn der Wurf unterwegs ist, wird kein Eingabepanel (die Eingabefelder, deren Beschriftung und der Wurfbutton) angezeigt.

Über die **Enter**-Taste oder über Klick auf den Wurf-Button wird der Wurf ausgelöst.

Das Framework stellt Operationen bereit, die Sie nutzen können, um diese Funktionalität zu implementieren.

Wurfverhalten—8 Punkte Der Wurf beginnt knapp über bzw. versetzt über dem Gorilla (möglichst realistisch). Das Wurfobjekt fliegt eine Parabelkurve.

Für einen Wurf von links nach rechts lässt sich die Position (x, y) des Wurfobjekts in Abhängigkeit der Zeit t, der Startposition (x_0, y_0) der Startgeschwindigkeit v und des Anfangswinkels α folgendermaßen berechnen:

$$
\begin{aligned}
v_x &= \cos(\alpha) \cdot v \\
v_y &= \sin(\alpha) \cdot v \\
x &= x_0 + (v_x \cdot t) \\
y &= y_0 - (v_y \cdot t) + (\frac{1}{2} \cdot g \cdot t^2)
\end{aligned}
$$

Der Wurf kann oben aus Bildschirm fliegen; der Zug ist erst beendet, wenn der Gegner oder die Skyline getroffen wurde, oder das Wurfobjekt den Bildschirm unten verlassen hat.

Damit die Flugkurve realistisch wird, sollten Sie das *delta* der Update-Methode von Slick, um zu t zu gelangen mit einem konstanten Faktor skalieren. Diesen "time scaling"-Faktor müssen sie im Testadapter über eine Methode bereitstellen, damit die Flugkurve exakt testbar wird.

Erkennen Zugende: Skyline getroffen—6 Punkte Es wird erkannt, dass ein Hochhaus getroffen wurde. Dabei verschwindet die Banane und es entsteht ein Schaden am Gebäude. Wahlweise können Sie an der Stelle des Einschlages zusätzlich eine Explosion anzeigen lassen.

Nach dem Einschlag wird das Eingabepanel des_der nächsten Spielers_Spielerin freigeschaltet, damit diese_r den eigenen Wurf parametrisieren kann.

Erkennen Zugende: Banane verlässt Bildschirm—4 Punkte Verlässt die Banane den Bildschirm nach unten, nach links oder nach rechts, so ist der_die nächste Spieler_in an der Reihe. Dafür ist das entsprechende Eingabepanel anzuzeigen.

Erkennen Zugende: Gegner getroffen—6 Punkte Es wird erkannt, wenn der Gegner getroffen wurde. Dann verschwindet die Banane und der gegnerische Gorilla wird zerstört bzw. verschwindet. Wahlweise können Sie eine Explosion rendern und/oder die Zerstörung des Gorillas animieren.

Wahlweise kann der gewinnende Gorilla jubeln.

Nach diesen Aktionen erscheint ein Fenster: Dort wird dem_der Gewinner_in gratuliert. Mit einem Klick auf einen "OK"-Button können die Spieler_innen in das Hauptmenü zurückkehren.

3.5 Ausbaustufe I

Die Ausbaustufe I erweitert die minimale Ausbaustufe. **Alle Anforderungen der minimalen Ausbaustufe gelten weiterhin, sofern sie unten nicht explizit geändert oder erweitert werden.**

Bei Implementierung dieser Ausbaustufe können insgesamt 75 aus 100 möglichen Punkten erreicht werden. Die öffentlichen Testfälle für diese Ausbaustufe befinden sich in der Klasse `GorillasTestsuiteExtended1`.

Öffentliche Tests der Ausbaustufe 1 sind erfolgreich—0 Punkte Die öffentlichen Testfälle der Klasse `GorillasTestsuiteExtended1` werden fehlerfrei absolviert. Beachten Sie, dass Sie für den Ablauf der Tests die Klasse `GorillasTestAdapterExtended1` implementieren müssen!

Sinnvolle Modellierung—5 Punkte Wenn Sie das vorgegebene *eea*-Framework verwenden, dann achten Sie bei Ihrer Implementierung auf die sinnvolle Verwendung von Entitäten, Ereignissen und Aktionen. Verwenden Sie Packages zur Strukturierung Ihres Codes. Ihr_e Tutor_in wird im Rahmen des Code-Review bewerten, wie gut Ihnen die Umsetzung gelungen ist.

Andernfalls strukturieren Sie Ihr Programm nach dem Prinzip *MVC („Model View Controller“)*, siehe T20.42-47. Führen Sie eine sinnvolle Einteilung in Pakete (*package hierarchy*) ein. Erstellen Sie ein Klassendiagramm in UML, welches die Struktur des Programms wiedergibt, und markieren Sie darin jeweils die Klassen der Bereiche Model, View und Controller. Ihr_e Tutor_in wird sich das Klassendiagramm ansehen und ggf. Fragen hierzu stellen. Insbesondere beim Code-Review wird auf die Umsetzung des MVC-Prinzips geachtet.

Highscore-Liste—5 Punkte Es soll eine Highscore-Liste persistent gespeichert und aktualisiert (und daher auch eingelesen) werden. Das Dateiformat, das für die Highscore-Liste verwendet wird, bleibt Ihnen überlassen, muss aber die Endung `hsc` haben.

Der Highscore-Eintrag für eine_n Spieler_in sollte folgendes enthalten:

1. Den Namen des Spielers
2. Die Anzahl der insgesamt gespielten Runden
3. Die Anzahl der gewonnenen Runden
4. Der prozentuale Anteil der gewonnenen Runden an den insgesamt gespielten Runden
5. Die durchschnittliche Wurfgenauigkeit, d.h. die durchschnittlich benötigte Anzahl an Würfen pro Treffer

Die Highscore-Einträge sollen nach den folgenden Kriterien in der folgenden Reihenfolge (wichtigstes Sortierkriterium zuerst) sortiert werden:

1. Prozentsatz der gewonnenen Spiele (absteigend)
2. Durchschnittliche Wurfgenauigkeit, d.h. durchschnittliche benötigte Anzahl an Würfen für einen Treffer (aufsteigend)

Highscore GUI—5 Punkte Die Highscore-Liste soll in der GUI angezeigt werden, wobei die Einträge entsprechend der Sortierung der Highscore-Liste angezeigt werden sollen. Durch Drücken der **Escape**-Taste soll zurück ins Hauptmenü gewechselt.

In der Highscore-GUI sollten nur die 10 besten Spieler_innen angezeigt werden.

Beachten Sie, dass alle Testfälle ohne Benutzerinteraktion durchlaufen müssen.

Rundenbasiertes Spiel: Kartengenerator—5 Punkte Zu jeder Runde des Spiels soll eine neue, zufällig generierte Skyline erstellt werden. Die Hochhäuser der Skyline sollen rechteckig sein. Das Dach jedes Hochhauses muss einen Abstand von mindestens 100 Pixeln vom oberen Spielfeldrand haben. Die Gorillas sollen weiterhin wie in der minimalen Ausbaustufe platziert werden. Die zufällige Skyline besteht aus mindestens sechs Häusern.

Rundenbasiertes Spiel: Spiel bis 3 und Score-Anzeige—3 Punkte Der Spielstand wird während des Spielens richtig angezeigt. Das Spiel soll sofort beendet sein, wenn eine_r der beiden Spieler_innen drei Runden gewonnen hat.

Wurfparameter werden gespeichert—2 Punkte Innerhalb einer Runde werden die vorher eingegebenen Wurfparameter pro Spieler_in gespeichert, damit sich der_die Spieler_in diese nicht mühsam merken muss.

3.6 Ausbaustufe II

Die Ausbaustufe II erweitert Ausbaustufe I und die minimale Ausbaustufe. **Alle Anforderungen dieser Ausbaustufen gelten weiterhin, sofern sie unten nicht explizit geändert oder erweitert werden.**

Bei Implementierung dieser Ausbaustufe können insgesamt 90 aus 100 möglichen Punkten erreicht werden. Die öffentlichen Testfälle für diese Ausbaustufe befinden sich in der Klasse `GorillasTestsuiteExtended2`.

Öffentliche Tests der Ausbaustufe 2 sind erfolgreich—0 Punkte Die öffentlichen Testfälle der Klasse `GorillasTestsuiteExtended2` werden fehlerfrei absolviert. Beachten Sie, dass Sie für den Ablauf der Tests die Klasse `GorillasTestAdapterExtended2` implementieren müssen!

Wind—5 Punkte Das Wurfverhalten wird nun zusätzlich von Wind beeinflusst. Zu jeder Runde, also jeder neuen Karte/Skyline/Landschaft, wird zufällig festgelegt, wie stark der Wind von links oder rechts weht. Dies wird visualisiert (z. B. durch einen Windpfeil). Der Wind beeinflusst den Wurf realistisch.

Für einen Wurf von links nach rechts lässt sich die Position (x, y) des Wurfobjekts in Abhängigkeit der Zeit t, der Startposition (x_0, y_0) der Startgeschwindigkeit v und des Anfangswinkels α nun folgendermaßen berechnen:

$$
\begin{aligned}
v_x &= \cos(\alpha) \cdot v \\
v_y &= \sin(\alpha) \cdot v \\
x &= x_0 + (v_x \cdot t) + (\frac{1}{2} \cdot w_{scale} \cdot w \cdot t^2) \\
y &= y_0 - (v_y \cdot t) + (\frac{1}{2} \cdot g \cdot t^2)
\end{aligned}
$$

Die Variable w besteht aus einem Wert zwischen -15 und 15. Diese Variable sollten Sie, analog zu der Skalierung des *delta* der Update-Methode von Slick, mit dem Faktor w_{scale} so verändern, dass der Wurf realistisch vom Wind beeinflusst wird.

Diesen "wind scaling"-Faktor müssen Sie im Testadapter wiederum über eine Methode bereitstellen, damit die Flugkurve exakt testbar wird.

Dotzen—5 Punkte Das Wurfobjekt fällt nicht durch die untere Begrenzung des Spielfelds / des Fensters, sondern dotzt dort wie ein Ball. Erst wenn das Wurfobjekt nur noch wenig Geschwindigkeit hat, fällt es nach unten, was den Zug beendet.

Sonne—3 Punkte Im Himmel über der Skyline befindet sich eine Sonne, die erstaunt guckt, wenn das Wurfobjekt durch sie hindurch fliegt.

Spieler_innennamen werden gespeichert—2 Punkte Bei der Namensauswahl der Spieler_innen werden die zuletzt ausgewählten Namen angezeigt, so dass zwei Spieler_innen, die mehrere Spiele absolvieren, diese nicht ständig neu eingeben müssen.

3.7 Ausbaustufe III

Die Ausbaustufe III erweitert Ausbaustufe II, I und die minimale Ausbaustufe. **Alle Anforderungen dieser Ausbaustufen gelten weiterhin, sofern sie unten nicht explizit geändert oder erweitert werden.**

Bei Implementierung dieser Ausbaustufe können insgesamt 100 aus 100 möglichen Punkten erreicht werden. Die öffentlichen Testfälle für diese Ausbaustufe befinden sich in der Klasse `GorillasTestsuiteExtended3`.

Öffentliche Tests der Ausbaustufe 3 sind erfolgreich—0 Punkte Die öffentlichen Testfälle der Klasse `GorillasTestsuiteExtended3` werden fehlerfrei absolviert. Beachten Sie, dass Sie für den Ablauf der Tests die Klasse `GorillasTestAdapterExtended3` implementieren müssen!

Erdbeschleunigung kann verändert werden—4 Punkte Die Erdbeschleunigung kann über ein Einstellungsmenü verändert werden. Beispielsweise kann das Wurfverhalten verändert werden, so dass die Spieler nicht nur die Hochhäuser und die Gorillapositionen und den Wind,

sondern auch die veränderbare Erdbeschleunigung in ihre Parametrisierung mit einbeziehen müssen.

Spöttische Bemerkungen—4 Punkte Je nachdem wie gut oder schlecht der Wurf war, wird der_die an der Reihe gewesene Spieler_in durch spöttische (Text-)Nachrichten angestachelt. Die Nachrichten sollten sich klar auf die Güte des Wurfes beziehen, etwa darauf, ob der Wurf viel zu weit oder viel zu kurz war.

Anleitung—2 Punkte Zu guter Letzt sollte in einem Extra-Menü eine kurze Anleitung zum Spiel vorhanden sein, um das Spiel zu komplettieren.

3.8 Bonuspunkte

Sie können auch mehr als 100 Punkte erreichen sowie fehlende Punkte aus anderen Ausbaustufen ausgleichen, indem Sie weitere Funktionen implementieren, die in den Ausbaustufen nicht spezifiziert wurden. Die Bewertung ist dabei Sache der Tutor_innen und der Veranstalter_innen. Zur Orientierung stellen wir hier beispielhaft mögliche Erweiterungen mit Punktzahl vor, damit Sie wissen, was wir ungefähr erwarten.

Nutzung von Audio-Clips—1 Punkt Audio-Clips werden bei bestimmten Ereignissen abgespielt (z. B. einer gewonnenen Karte).

„About“-Fenster—1 Punkt Implementieren Sie eine About-Box, d.h. ein Fenster, das die Namen der beteiligten Mitglieder Ihrer Projektgruppe auflistet. Sie kennen diese Fenster sicher aus vielen bekannten Programmen. Seien Sie kreativ! :-)

Überraschen Sie uns—0 Punkte Sie können auch eigene Ideen zum Ausbau des Spiels einbringen. Die Bewertung mit Punkten ist dann Sache der Verstalter_innen und Tutor_innen. Die 0 ist also *nicht wörtlich zu nehmen*.

Ein paar „Standardideen“ sind etwa *Easter Eggs*, eine Statuskonsole oder fetzige Hintergrundmusik.

Hinweis: Der Code in den Testklassen **darf auf keinen Fall** zur Anzeige von Dialogfenstern führen. Dies darf auch bei inkorrekten Schritten nicht geschehen, da sonst eine Eingabe erfordert würde, die in den JUnit-Tests nicht automatisiert erfolgen kann. Zum Testen darf nicht der *org.newdawn.slick.AppGameContainer* werden, da dieser zur Anzeige von Dialogfenstern führt. Verwenden Sie stattdessen *de.tu_darmstadt.gdi1.gorillas.tests.TestAppGameContainer*.

Die Tests benutzen weiterhin nicht die Klasse *de.tu_darmstadt.gdi1.gorillas.main.Gorillas* sondern die Klasse *de.tu_darmstadt.gdi1.gorillas.test.setup.TestGorillas*. Diese Klasse erbt statt von *de.matthiasmann.twl.slick.TWLStateBasedGame* von *de.tu_darmstadt.gdi1.gorillas.test.setup.TWLTestStateBasedGame*. Sie müssen folglich exakt wie Sie in der Klasse Gorillas Ihre States hinzugefügt haben, dies auch in TestGorillas tun. TWLTestStateBasedGame verzichtet im Gegensatz zu TWLStateBasedGame auf das Rendern der TWL-GUI-Elemente.

Bitte achten Sie bei den Erweiterungen darauf, dass alte Funktionalität nicht verloren geht.

4 Materialien

Auf der Veranstaltungsseite stellen wir Ihnen einige Dateien zur Verfügung, die Sie für Ihre Lösung verwenden können. Dabei handelt es sich um vorgefertigte Karten und die öffentlichen Testfälle. Es wird Ihnen dringend nahe gelegt, auch unser bereitgestelltes Framework nutzen. Das Framework ist threadsicher!

Hinweis: Sie dürfen auch eine vollständig eigene Lösungen implementieren. Der dadurch entstehende Mehraufwand wird aber nicht durch Punkte gewürdigt. Ihre Lösung **muss** in jedem Fall mit den bereitgestellten Testklassen überprüfbar sein.

Neben der kurzen Beschreibung in diesem Dokument gibt Ihnen die JavaDoc-Dokumentation nützliche Hinweise zur Verwendung und den Parametern der vorhandenen Funktionen. Ein Blick in diese Quellen empfiehlt sich sehr—**am besten vor dem Gang zum_zur Tutor_in**, um. . .

- sich selbst und auch dem_der Tutor_in wertvolle Zeit zu sparen, da sich so unter Umständen banale Fragen von selbst lösen,
- Nerven zu sparen, da das Warten auf die Antwort der Tutor_innen Ihre Nerven beanspruchen wird :-),
- sich selbstständiges Arbeiten anzueignen.

4.1 Klasse *de.tu_darmstadt.gdi1.gorillas.main.Gorillas*

Gorillas erbt von *de.matthiasmann.twl.slick.TWLStateBasedGame*. *Gorillas* dient sowohl als Container für Spiele als auch zum Starten des gesamten Spiels. Sie müssen diese Klasse erweitern, um zusätzliche States hinzuzufügen. Lesen Sie sich dazu das *Drop of Water Tutorial* durch.

4.2 Die ersten Schritte

Sie sollten sich zunächst das *Drop of Water Tutorial* durchlesen, um das Konzept des Frameworks zu verstehen. Anschließend können Sie in `Gorillas` zusätzliche States (Fenster) definieren. Sie sollten mit einem Hauptmenü und einem Spielfenster beginnen.

Anschließend sollten Sie in der Gruppe diskutieren, wie Ihr Projekt strukturiert werden soll und Aufgaben in der Gruppe verteilen.

4.3 Basisereignisse und Basisaktionen

Das **eea**-Framework arbeitet mit **Entitäten**, wie der **ImageRenderComponent**, sowie **Events** und zugehörigen **Actions**. Die zwei Tabellen am Ende geben einen Überblick über die mitgelieferten Events und Actions. Das Neuzeichnen eines Fensters geschieht zusammen mit einem Frame.

5 Zeitplanung

Wir empfehlen Ihnen, **vor Beginn der Bearbeitung** zuerst eine für Ihre Gruppe realistische Ausbaustufe auszuwählen und die Aufgabe dann in parallel bearbeitbare Teilaufgaben zu zerlegen, die

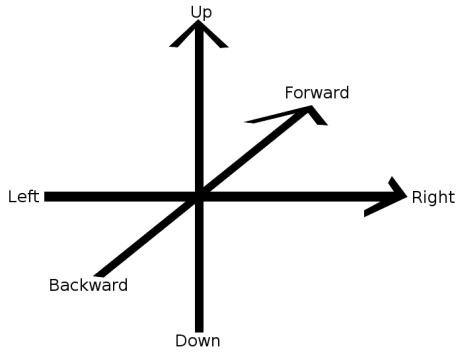

Abbildung 3: Es gibt zahlreiche Aktionen, die eine Bewegung ausdrücken.

Sie auf die einzelnen Gruppenmitglieder verteilen. Über zentrale Elemente wie die grundsätzlichen Klassennamen und die Vererbungshierarchie sollten Sie sich in der Gruppe einigen, um spätere Diskussionen zu vermeiden.

Erstellen Sie einen schriftlichen Zeitplan, wann Sie welche Aufgabe abgeschlossen haben möchten. Dabei ist es wichtig, den aktuellen Projektstand regelmäßig kritisch zu überprüfen. Ein solches geplantes Vorgehen vermeidet Stress (und damit unnötige Fehler) beim Abgabetermin.

Eine Zeitplanung für die minimale Ausbaustufe könnte etwa wie in Tabelle 3 auf Seite 21 gezeigt aussehen. Eine denkbare Gesamteinteilung für alle Ausbaustufen wird in Tabelle 4 auf Seite 21 gezeigt.

Bitte beachten Sie, dass **alle Zeiten stark von Ihrem Vorwissen und Ihren Fähigkeiten abhängen**, so dass die Zeiten nicht als „verbindlich“ betrachtet werden dürfen!

Tipp: Wenn Sie eine Ausbaustufe fertig implementiert haben, sollten Sie einen Gang zum_zur Tutor_in nicht scheuen, ehe Sie sich gleich an die nächste Stufe heranwagen. Fragen Sie Ihre_n Tutor_in nach eventuellen Unklarheiten, falls Sie z. B. inhaltlich die Aufgabenstellung nicht verstanden haben. Es ist Aufgabe der Tutor_innen, Fragen zu beantworten, also nutzen Sie dieses Angebot.

Wichtig: „Sichern“ Sie stabile Zwischenergebnisse, etwa indem Sie ein Versionskontrollverfahren wie *SVN* oder *Git* nutzen. Tipps dazu finden Sie im Portal. Die einfache Variante ist es, regelmäßig eine JAR-Datei mit den Quellen (!) anzulegen, die Sie jeweils je nach erreichtem Status „passend“ benennen. Dann haben Sie immer eine Rückfallmöglichkeit, falls etwa nach einem Code-Umbau nichts mehr funktioniert.

6 Anpassungen für Studierende im Studiengang CE

Die Anforderungen für Studierende im Studiengang CE ersetzen Teile der Ausbaustufe I-III, wie unten erläutert. Dabei sind zwei Aspekte umzusetzen: eine *realistische Flugbahn unter Berücksichtigung der Luftreibung* sowie *(semi-)dynamische Windböen*.

Achtung: die Anforderung „Wurfverhalten“ der minimalen Ausbaustufe ist zusätzlich *unverändert*

zu implementieren, da sonst zahlreiche Tests scheitern. Gehen Sie dazu so vor, dass in Ihrem Spiel *zwei* Funktionen zur Berechnung der Flugbahn existieren. Das *Spiel* selbst soll die realistischere, aber komplexere Formel nutzen, während die *Tests* auf das vereinfachte „Wurfverhalten“ zurückgreifen.

Die folgenden Anforderungen gelten für Studierende im Studiengang Computational Engineering (CE) sowie für gemischte Gruppen, an denen CE-Studierende beteiligt sind. Dabei gehören die ersten beiden Punkte zusammen:

Die parabolische Flugbahn soll die Luftreibung wie folgt berücksichtigen:

$$F = 0.5 \cdot \rho \cdot C \cdot A \cdot V^2$$

mit

- A = Fläche der Banane
- ρ = Dichte (1.293 $\frac{kg}{m^2}$) bei 0° Celsius und 1 Atmosphere Luftdruck)
- C = 2.1
- V = Geschwindigkeit relativ zur Luftgeschwindigkeit, d.h. $v(t) - v_{luft}(t)$.

Auf die Berücksichtigung der Rotation der Banane kann verzichtet werden.

Die Berechnung des Bananenflugs soll entweder das Runge-Kutte-Verfahren[2] oder symplektische Euler-Integration wie folgt verwendet werden:

$$\begin{aligned} a(t) &= F(v(t), x(t), t) \\ v(t+1) &= v(t) + dt \cdot a(t) \\ x(t+1) &= x(t) + dt \cdot v(t+1) \end{aligned}$$

Windböen wehen mit einer eigenen Luftgeschwindigkeit $v_{luft}(t) = v_l \cdot sin(2 \cdot \pi \cdot t/D)$, wobei v_l und D Zufallszahlen sind. Bestimmen Sie für die Zufallszahlen durch Ausprobieren und/oder entsprechende Überlegungen eine passende Begrenzung, um unrealistisch starke Winde oder unglaubwürdige Windrichtungen möglichst auszuschließen.

Hinweis: Die Windböen sollen innerhalb eines Spielzugs—also von dem Zeitpunkt des Spielerwechsels bis zum Ende der Flugbahn der geworfenen Banane—*konstant* bleiben, wechseln also erst mit dem nächsten Spielzug. Dies ist zwar nur bedingt realistisch, ermöglicht aber die „Spielbarkeit“. Bei (realistischen) dynamischen Windböen, die auch während der Eingabe der Flugparameter und während des Flugs ihre Stärke und/oder Richtung variieren können, können an sich präzise ausgerechnete Würfe das Ziel verfehlen—und gleichzeitig ziemlich falsch berechnete Werte „zufällig treffen“. Damit wäre Gorillas effektiv unspielbar, da der Zufall eine zu große Rolle spielen wwürde.

[2] siehe `http://de.wikipedia.org/wiki/Runge-Kutta-Verfahren`

Die beiden ersten Aspekte („Parabolische Flugbahn mit Luftreibung und realistischer Berechnung“ ersetzen insgesamt die beiden Anforderungen „Erdbeschleunigung kann verändert werden“ und „Spöttische Bemerkungen“ aus Aufbaustufe III und ergeben daher 8 Punkte.

Die Anforderung „Windböen“ ersetzt für CE-Gruppen die Anforderung „Sonne“ und ergibt damit 3 Punkte.

„Nicht-CE-Gruppen“, die diese Aufgaben (korrekt) bearbeiten, erhalten die genannten Punktzahlen als *Bonuspunkte*. Analog erhalten CE-Gruppen, die die „gestrichenen“ Aufgabenteile (korrekt) bearbeiten, die entsprechende Punktzahl als Bonuspunkte.

Bitte beachten Sie, dass die obigen Aufgaben *nicht* Bestandteil der Testfälle sind. Weicht das von Ihnen berechnete Verhalten von der (einfacheren) allgemeinen Formel ab, werden die entsprechenden Tests entsprechend fehlschlagen. Dadurch müssen Sie nicht in Panik geraten; ihr_e Tutor_in wird Ihre Implementierung dieser Aspekte im Code begutachten und bewerten. Denken Sie an den Tip mit den „zwei Varianten“, um Fehler in den Tests zu vermeiden!

7 Liste der Anforderungen

Anforderungen an das Projekt Seite

Pünktliche Abnahme (0 Punkte) - Stufe: Minimal . . . 7
Compilierbares Java (0 Punkte) - Stufe: Minimal . . . 7
Nur eigener Code (0 Punkte) - Stufe: Minimal . . . 7
Vorbereitung für automatisierte Tests (0 Punkte) - Stufe: Minimal . . . 7
Öffentliche Tests sind erfolgreich (0 Punkte) - Stufe: Minimal . . . 8
Grafische Benutzerschnittstelle (3 Punkte) - Stufe: Minimal . . . 8
Spielmenü (2 Punkte) - Stufe: Minimal . . . 8
Neues Spiel starten (2 Punkte) - Stufe: Minimal . . . 8
Eingabe und Anzeige der Spielernamen (6 Punkte) - Stufe: Minimal . . . 8
Grafische Umsetzung der Objekte (4 Punkte) - Stufe: Minimal . . . 8
Platzierung der Gorillas (3 Punkte) - Stufe: Minimal . . . 8
Eingabe der Wurfparameter (6 Punkte) - Stufe: Minimal . . . 8
Wurfverhalten (8 Punkte) - Stufe: Minimal . . . 9
Erkennen Zugende: Skyline getroffen (6 Punkte) - Stufe: Minimal . . . 9
Erkennen Zugende: Banane verlässt Bildschirm (4 Punkte) - Stufe: Minimal . . . 9
Erkennen Zugende: Gegner getroffen (6 Punkte) - Stufe: Minimal . . . 10
Öffentliche Tests der Ausbaustufe 1 sind erfolgreich (0 Punkte) - Stufe: 1 . . . 10
Sinnvolle Modellierung (5 Punkte) - Stufe: 1 . . . 10
Highscore-Liste (5 Punkte) - Stufe: 1 . . . 10
Highscore GUI (5 Punkte) - Stufe: 1 . . . 11
Rundenbasiertes Spiel: Kartengenerator (5 Punkte) - Stufe: 1 . . . 11
Rundenbasiertes Spiel: Spiel bis 3 und Score-Anzeige (3 Punkte) - Stufe: 1 . . . 11

Wurfparameter werden gespeichert (2 Punkte) - Stufe: 1 11
Öffentliche Tests der Ausbaustufe 2 sind erfolgreich (0 Punkte) - Stufe: 2 11
Wind (5 Punkte) - Stufe: 2 11
Dotzen (5 Punkte) - Stufe: 2 12
Sonne (3 Punkte) - Stufe: 2 12
Spieler_innennamen werden gespeichert (2 Punkte) - Stufe: 2 12
Öffentliche Tests der Ausbaustufe 3 sind erfolgreich (0 Punkte) - Stufe: 3 12
Erdbeschleunigung kann verändert werden (4 Punkte) - Stufe: 3 12
Spöttische Bemerkungen (4 Punkte) - Stufe: 3 13
Anleitung (2 Punkte) - Stufe: 3 13
Nutzung von Audio-Clips (1 Punkte) - Stufe: Bonus 13
„About“-Fenster (1 Punkte) - Stufe: Bonus 13
Überraschen Sie uns (0 Punkte) - Stufe: Bonus 13

Konstruktor	**Beschreibung**
CollisionEvent()	Dieses Event wird (mit jedem Frame) ausgelöst, wenn sich zwei Entitäten überlappen.
KeyPressedEvent(String id, Integer... key)	Dieses Event wird ausgelöst, wenn die Taste gerade erstmalig nach unten gedrückt wurde.
KeyDownEvent(String id, Integer... key)	Dieses Event wird ausgelöst, wenn die Taste aktuell unten gehalten wird.
LeavingScreenEvent()	Dieses Event wird ausgelöst, wenn die an diesem Event interessierte Entität die Spielfläche verlassen hat.
LoopEvent(String id)	Dieses Event wird mit jedem Frame ausgelöst. Dieses Event eignet sich also dafür, in jedem Frame eine gewisse Aktion auszuführen.
MouseClickedEvent()	Dieses Event wird ausgelöst, wenn die Maus auf die an dem Event interessierte Entität geklickt hat.
MouseEnteredEvent()	Dieses Event wird ausgelöst, wenn die Maus über die an dem Event interessierte Entität fährt.
MovementDoesntCollideEvent(float speed, IMovement move)	Dieses Event wird ausgelöst, wenn die Bewegung (Action) keine Kollision mit einer anderen Entität verursacht.

Tabelle 1: Basisereignisse des **eea**-Frameworks aus dem package *eea.engine.events.basicevents*

Konstruktor	Beschreibung
ChangeStateAction(int state)	Diese Action wechselt in einen Zustand (State) mit der State-ID **state**. Ist der State schon einmal initialisiert worden, so wird die **init**-Methode nicht erneut aufgerufen.
ChangeStateInitAction(int state)	Diese Action wechselt in einen State mit der State-ID state. Auch wenn der State schon einmal initialisiert worden ist, wird die **init**-Methode erneut aufgerufen.
DestroyEntityAction()	Diese Action zerstört die Entität bei Eintreten eines gewissen Events.
*MoveBackwardAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Rückwärtsbewegung entgegen der Blickrichtung aus.
*MoveDownAction(float speed()**	Diese Action führt mit einer Geschwindigkeit **speed** eine Bewegung nach unten aus.
*MoveForwardAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Vorwärtsbewegung in Blickrichtung aus.
*MoveLeftAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Bewegung nach links aus.
*MoveRightAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Bewegung nach rechts aus.
*MoveUpAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Bewegung nach oben aus.
*RotateLeftAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Drehung nach links aus.
*RotateRightAction(float speed)**	Diese Action führt mit einer Geschwindigkeit **speed** eine Drehung nach rechts aus.
QuitAction()	Diese Action beendet das laufende Spiel.
RemoveEventAction()	Diese Action zerstört ein Event.
SetEntityPositionAction(Vector2f pos)	Diese Action setzt die Position einer Entität neu.

Tabelle 2: Basisaktionen des **eea**-Frameworks aus dem package *eea.engine.actions.basicactions.* Die mit einem * markierten Aktionen sind für das Spiel Gorillas nicht von Bedeutung. Um den Wurf eines Objektes zu simulieren, empfiehlt es sich eine eigene MoveAction zu implementieren. Die Richtungen werden verdeutlicht durch Abbildung 3 auf Seite 15.

Aspekt	**Aufwand (ca.)**
Einarbeitung in die Aufgabenstellung und Vorlagen, Lesen des Tutorials	1.5 Tage
Einigung auf die grundsätzliche Klassen- und Packagehierarchie	0.25 Tage
Entwicklung der grafischen Benutzerschnittstelle (Spielfenster und Menüfenster)	0.25 Tag
Neues Spiel per Tastendruck	0.1 Tag
Eingabe und Anzeige der Spielernamen	0.5 Tag
Umsetzung der grafischen Objekte	0.1 Tag
Eingabe der Wurfparameter	0.3 Tag
Wurfverhalten	0.5 Tag
Erkennen Zugende: Skyline getroffen	0.25 Tag
Erkennen Zugende: Banane verlässt Bildschirm	0.1 Tag
Erkennen Zugende: Gegner getroffen	0.25 Tag

Tabelle 3: Zeitplanung für die Bearbeitung „nur“ der minimalen Ausbaustufe. Die Aufgaben werden in der Regel parallel von einzelnen oder mehreren Gruppenmitgliedern bearbeitet.

Stufe	**Geschätzter Zeitumfang**
Minimale Ausbaustufe	4 - 5 Tage
Ausbaustufe I	3 - 4 Tage
Ausbaustufe II	2 - 3 Tage
Ausbaustufe III	1 - 2 Tage
Bonusaufgaben	Falls hier von Anfang an ein *starkes* Interesse bestehen sollte, sollten Sie sich am besten vom ersten Tag an Gedanken machen und möglichst zügig anfangen...

Tabelle 4: Zeitplanung für die Bearbeitung aller Ausbaustufen mit 4 Personen