Serious Games
Sommersemester 2021
Praxisübung

Unity3d
Abgabe: **14.06.2021 23:59 Uhr**

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1 AllgemeineInformationen

Willkommen bei der praktischen Bonusübung im Sommersemester 2021. Sie werden in Teams von 2-4 Studierenden ein kleines Spiel selbst implementieren. Ihnen stehen dazu 3D- und Soundassets in Moodle zur Verfügung. Diese Assets dürfen wir Ihnen zur Verfügung stellen. Beachten Sie, dass Sie diese Assets nicht für eigene, kommerzielle Projekte verwenden dürfen. Sobald Sie in Moodle dem zugestimmt haben, erhalten Sie Zugriff auf die Assets.

Die Aufgaben sind darauf konzipiert, dass Sie jede Woche 10 Bonuspunkte erhalten können und 5 Wochen Bearbeitungszeit haben (insgesamt 50 Punkte). Damit Sie flexibler sind, erhalten Sie bereits sämtliche Materialien und Aufgabenstellungen und können anschließend die Bearbeitungszeit selbst organisieren. Das Projekt muss bis spätestens **14.06.2021 23:59 Uhr** abgegeben sein.
Um das Projekt abzugeben haben Sie 3 Möglichkeiten:

- Laden Sie eine .zip Datei in Moodle hoch, die Ihr Projekt beinhaltet. Um die Dateigröße zu minimieren können Sie vor dem Verpacken den Ordner Library sowie alle Ordner und Dateien, die von Ihrer IDE angelegt wurden, löschen (Bei Visual Studio Code wären das .vscode sowie die .sln Dateien).
- Laden Sie die Projektdatei auf Plattformen wie Dropbox, Google Drive, etc. hoch. In Moodle laden Sie eine Textdatei mit einem Link (Der uns einen Download erlaubt) dorthin hoch. Wir werden das Änderungsdatum, das in Ihrem Service angezeigt wird überprüfen und es muss vor dem Abgabedatum in Moodle sein (**14.06.2021 23:59 Uhr**).
- Laden Sie Ihr Projekt auf GitHub.com (Wenn nicht eh bereits geschehen). Laden Sie „Serious Games KOM Lehre“ zu Ihrem Projekt ein und erstellen Sie einen Release in Ihrem Repository. Der Zeitstempel des Releases muss selbstverständlich ebenfalls vor dem Abgabedatum in Moodle sein (**14.06.2021 23:59 Uhr**).

Wir werden im Zeitraum **15.06. – 28.06.2021** Testattermine anbieten. Melden Sie sich als Gruppe für einen solchen an und erscheinen Sie vollzählig (Digital, Zoom). während diesem Testattermin werden Sie kurz erklären, welche Aufgaben Sie bearbeitet haben. Zu diesen Aufgaben werden Sie anschließend gefragt, wie Sie die Aufgabe realisiert haben. Ein Testat sollte nicht über 15-20 Minuten dauern und wir wählen aus, wer antworten darf.

Das Spiel, das Sie implementieren werden, ist folgendes:
Der Spieler übernimmt einen Farmer.
Auf diesen Farmer stürmen Tiere zu.
Der Farmer muss rechtzeitig das korrekte Essen auf die Tiere werfen (Die Tiere "füttern").
Erreichen die Tiere ungefüttert den Farmer, verliert er das Spiel.

Kleine Anmerkungen zu den Aufgabenstellungen:

- Bei vielen Aufgabenstellungen sind Sie frei in der Umsetzung, wenn lediglich die Funktionalität des Spiels beschrieben ist. Bspw. Können Sie im Abschnitt *Die Kamera folgt dem Spieler* selbst wählen, welche Kameraperspektive Sie wählen möchten. Top-Down, First-Person, Third-Person,... ist alles erlaubt, solange die Kamera dem Spieler folgt.
- Es ist Absicht, dass nicht alle Komponente und Funktionalitäten, die hier abverlangt werden, in der Vorlesung erklärt wurden. Um bspw. Sound abzuspielen müssen Sie sich selbst das Component *AusioSource* anschauen.
- Falls es bisher nicht oft genug erwähnt wurde: Das Abgabedatum ist **14.06.2021 23:59 Uhr**.

2 Aufgaben

2.1 Woche 1 [10]

1. Erstellen Sie ein neues Unity Projekt in der Version 2020.3.X. (LTS). Sie dürfen eine beliebige Renderpipeline wählen. Die Assets stehen Ihnen in der Build-In und der URP zur Verfügung. Möchten Sie das Projekt mit der HDRP bearbeiten, müssen Sie die Assets selbst konvertieren. Importieren Sie die zur Verfügung gestellten Assets in Ihr Projekt. [1]
   NOTE: Abgaben in einer anderen Version werden NICHT bewertet!

2. Erstellen Sie eine neue Szene (Die Sie *Game* nennen) und platzieren Sie dort eine Ebene (3D Object -> Plane) [1]
3. Wählen Sie eine Humanoide Gestalt für Ihren Spieler (Also einen der Farmer aus *Course Library/Humans*). Implementieren Sie ein System, womit der Spieler sich per Tastatur-Input bewegen kann (Animationen sind kein verpflichtender Teil der Übung, was nicht heißt, dass Sie den Charakter nicht animieren dürfen). Platzieren Sie den Charakter auf der Ebene. [1] PlayerController放在人物模型顶层
4. Die Kamera folgt dem Spieler. [1]
5. Der Farmer darf einen begrenzten Spielbereich nicht verlassen können. [1]
6. Der Farmer schaut stets in entweder Bewegungsrichtung **oder** Mausrichtung. (Sie wählen ein Verhalten aus) [2]
7. Der Spieler kann ein Essensobjekt in eine Richtung werfen. Nach dem Wurf bewegt sich das Essen nur noch gerade aus. Es bietet sich an, dass das Essen in Richtung Mousecursor geworfen wird. [2] https://forum.unity.com/threads/throwing-objects.521309/
8. Bis hierher treten keine Glitches/Bugs auf. [1]

with collider, throw will be no use! https://www.youtube.com/watch?v=bA12WEA5MLo

2.2 Woche 2 [10]

1. Erstellen Sie 2 neue Szenen. Eine Szene, die das Hauptmenü darstellt und eine Szene, die den Endscreen nach dem Spiel darstellt. [1]
2. Vom Hauptmenü soll man das Spiel starten können. [1] https://www.youtube.com/watch?v=Dv6Tq1WY6rY
3. Vom Endscreen aus soll man ins Hauptmenü kommen können. [1] build settings add new scene
4. Vom Spiel aus gelangt man in den Endscreen (Hier genügt noch einfach einen Input/Button zu definieren. Es soll nur die Szenentransition bereits funktionieren) [1]
5. platzieren Sie mindestens eines der Tiere in der Szene. [0], die "Tierabhängigen"Punkte giebt es später)
6. Um Ressourcen zu sparen, soll sich das geworfene Essen selbst zerstören, wenn der Spieler kein Tier trifft. Entweder nach einer gewissen Zeit, oder wenn es den Spielraum verlässt. [1]
7. Trifft ein Essen auf ein Tier, wird das Essen zerstört. [1]
8. Trifft ein Essen auf ein Tier, wird das Essen zerstört und das Tier ebenfalls, **wenn** es das „korrekte“ essen ist. Es ist also möglich einzustellen, welches Essen für welches Tier geeignet ist. [2]
9. Startet die *Game*scene wird zu erst ein Fenster angezeigt, das die Steuerung erklärt. Drückt der Spieler auf einen Button (*Starten*, *Verstanden* o.Ä.), verschwinden das Fenster und das Spiel startet. [1] Playcontroller judge if clicking on the UI element
10. Bis hierher treten keine Glitches/Bugs auf. [1]

2.3 Woche 3 [10]

1. Außerhalb des Spielraumes spawnen nun Tiere über Zeit. [1] https://jingyan.baidu.com/article/3f16e0031b2a5f2591c10395.html
2. Die Tiere bewegen sich auf den Spieler zu (egal wo er sich gerade befindet) [1]
3. Die Tiere schauen immer in Richtung des Spielers [1]
4. Die Spawnrate der Tiere nimmt über Zeit zu. [1]
5. Der Spieler besitzt „Lebenspunkte“. Kollidiert der Spieler mit einem Tier, verliert er einen Punkt. [1]
6. Die aktuellen Lebenspunkte werden im UI angezeigt. [1]
7. Bleiben nur noch 0 Lebenspunkte über, endet das Spiel (Transition in den Endscreen). Hier können Sie nun die vorige Implementation der Szenentransition in den Endscreen entfernen. [1]
8. Man kann im Editor einstellen, welche Tiere spawnen können. Es können mehrere Tiere eingestellt werden, dann ist es Zufall, welches Tier gespawnt wird. [2]
9. Bis hierher treten keine Glitches/Bugs auf. [1]

Collision to detect, one of the objects should be rigidbody

2.4 Woche 4 [10]

1. Es ist möglich, neue Tiere und neues Essen zu erstellen, **ohne eine Coding-IDE zu öffnen** (Designer-Friendly-Programming).
   Konkretisierung nach Nachfrage: *Gemeint ist der Vorgang das Spiel um ein Tier zu erweitern.*
   Beispiel: Das Spiel enthält derzeit Pferde und Füchse. Das Team entscheidet sich, nun auch Strauße hinzuzufügen.
   Ein Gamedesigner entwirft nun einen Strauß, das dazugehörige Essen und definiert, dass das eben neu erstellte Essen zum Strauß gehört.
   Während diesem Vorgang muss der Designer keine Coding-IDE öffnen, trotzdem kann der Strauß anschließend der Szene/einem Spawner hinzugefügt werden. [3]
2. Der Spieler kann entscheiden, welches Essen er werfen möchte (Wie er das entscheidet, ist Ihnen überlassen). [1]
3. Im UI sieht der Spieler welches Essen er wirft, bzw. wie er welches Essen wirft. [1]

4. Es gibt mindestens 4 Tiere und 2 unterschiedliche Essensvarianten [1]
5. Es gibt eine Score, die sich automatisch über Zeit erhöht. [1]
6. Die Score wird ebenfalls erhöht, wenn der Spieler ein Tier korrekt „füttert“. [1] noch nicht fertig gemacht
7. Die Score wird im UI angezeigt. [1]
8. Bis hierher treten keine Glitches/Bugs auf. [1]

2.5 Woche 5 [10]

1. Schauen Sie sich das Konzept „Object-Pooling“ an. Das Essen soll so realisiert werden, dass es aus einem Object-Pool entnommen wird. https://github.com/Rfrixy/Generic-Unity-Object-Pooler
   Grob: Das Essen wird nicht mehr zerstört, sondern nur deaktiviert und kann wiederverwendet werden. [2]
2. Implementieren Sie mindestens 2 Schwierigkeitsgrade. Ein Schwierigkeitsgrad ändert die Bewegungsgeschwindigkeit der Tiere. [1]
3. Die Schwierigkeitsgrade sind vom Menü aus wählbar [2]
4. Im Endscreen wird die erreichte Score angezeigt. [1]
5. Im Endscreen wird ebenfalls die bisherige Highscore angezeigt (Kann identisch mit der erreichten Punktzahl sein, wenn der Spieler die Highscore geknackt hat. [1]
6. Bei jedem Wurf wird ein Soundeffekt abgespielt. [1]
7. Ein Hintergrundsound wird in einer Loop abgespielt. [1]
8. Das Spiel ist Glitch-/Bugfrei. [1]