LEHRSTUHL FÜR MEDIENINFORMATIK
Institut für Information und Medien, Sprache und Kultur
Universität Regensburg

Studienleistung I: Labyrinth Runner

PROTOTYPING IN UNITY; SOMMERSEMESTER 2016

Abgabe: Dienstag 24. Mai 2015 23:55 via Grips / GIT

Beschreibung

In dieser Studienleistung sollen Sie einen Prototyp in Unity3D selbstständig umsetzen. Damit sollen die grundlegenden Fertigkeiten mit der Game Engine wiederholt werden.

Implementieren Sie ein kleines Spiel, in dem es darum geht, dass der Spieler den Ausgang aus einem Labyrinth möglichst schnell findet. Dazu muss er mit Türen interagieren können und diese öffnen. Der Spieler gewinnt, sobald ein gewisser Punkt in der Szene erreicht wurde. Dabei muss der Spieler jedoch aufpassen, dass er nicht in Bomben tritt, die ihm Lebenspunkte abziehen könnten. Hat der Spieler keine Lebenspunkte mehr, verliert man das Spiel und muss wieder von vorne beginnen.

Sie haben die Studienleistung bestanden, sobald ein funktionierender Prototyp im Git-Repository (+GRIPS) vorliegt und das geforderte Verhalten implementiert ist.

Viel Spaß! ☺

Arbeitsschritte

Bewegung der Spielfigur

Der Spieler muss sich durch den Raum bewegen und mit der Umgebung interagieren können. Die Kamera bleibt statisch und betrachtet das Geschehen aus einer festen Richtung. Der Spieler wird Lebenspunkte und eine maximale Geschwindigkeit haben. Die maximale Geschwindigkeit kann über eigene externe Variable gesteuert werden.

Öffnen Sie die Klasse „PlayerInputManager“ und erweitern Sie dieses Script um die folgende Funktionalität:

- Implementieren Sie die Steuerung: mit den Tasten W und S bewegen Sie die Spielfigur nachvorne bzw. rückwärts und mit „A“ bzw. „D“ drehen Sie die Figur. Bitte beachten Sie, dass Sie die Implementierung auf der integrierten Physiksimulation aufbaut (kontrollieren Sie den Rigidbody des Spielers und nicht die Transformation direkt, da Sie sonst keine Kollisionserkennung nutzen können).
- Die Geschwindigkeit des Spielers soll über eine Variable im Ressourcenmanager des Spielers gesteuert werden (z.B. soll die Figur langsamer laufen können, sobald die Spielfigur wenige Lebenspunkte hat)
- Implementieren Sie eine Methode, um die Position des Spielers an eine bestimmte Position zu setzen (z.B. beim Respawn)

Ressourcenverwaltung und Spielerstatus

Nun kann der Spieler die Welt erkunden. Im nächsten Schritt statten wir den Spieler mit seinen Ressourcen aus.

Öffnen Sie die Klasse „PlayerManager" und erweitern Sie dieses Script um die folgende Funktionalität:

- Implementieren Sie die Lebenspunkte und die Geschwindigkeit des Spielers sowie Möglichkeiten zur Veränderung der Werte.
- Ferner verliert der Spieler, sobald er keine Lebenspunkte mehr hat. Implementieren Sie daher eine Funktionalität, die meldet, dass der Spieler tot ist und ein Respawn eingeleitet wird. (Hierzu wäre eine globale Instanz eines GameManagers sinnvoll)

Türen öffnen

Der Ausgang ist hinter verschiedenen Türen versteckt, die wir im Verlauf des Spiels öffnen müssen. Dazu benötigen die Türen an sich eine eigene Komponente.

Implementieren Sie ein Script „Door.cs", welches das folgende Verhalten implementiert:

- Wird ein Boolean „isOpen" auf „true" gesetzt, soll sich die Tür soweit in den Boden bewegen, dass diese nicht mehr sichtbar ist. Ändert sich der Wert wieder auf „false", soll die Türe auch wieder in die Ausgangssituation sich zurückbewegen (Achtung: die Tür soll animiert sein und nicht plötzlich verschwinden).
- Aktualisieren Sie nun alle Türen in der Szene und statten Sie die Türen mit dem neuen Script aus.

Um diese Türen im Labyrinth öffnen zu können, muss der Spieler mit den Konsolen in der Spielwelt interagieren können:

Legen Sie ein Prefab „TriggerConsole" mit einer Collider-Komponente und einem eigenen Script an.

Legen Sie ein Script an „DoorTrigger.cs", welches das folgende Verhalten implementiert:

- *Sobald der Spieler die eigene Trigger-Zone betreten hat, soll der Spieler mit Hilfe eines Tastendrucks eine referenzierte Türe öffnen können.*
- *Eine Grafik/der Button zum Drücken soll im User Interface angezeigt werden, sobald der Spieler diese Trigger-Zone betreten hat bzw. ausgeblendet werden, sobald der Spieler diese Zone wieder verlässt.*
- *Ein Sound soll abgespielt werden, sobald der Spieler die Trigger-Zone betreten hat.*

Andere Trigger

Implementieren Sie einen Trigger (=Bombe), welcher Lebenspunkte vom Spieler abzieht, sobald der Spieler den Trigger betritt. (Nehmen Sie zur Visualisierung verschiedene Hilfsassets wie Würfel oder dergleichen). Hat der Spieler einmal diesen Trigger ausgelöst, wird der Trigger deaktiviert.

Implementieren Sie einen Trigger, der als Siegbedingung gilt (Das Spiel endet sobald der Nutzer den Trigger betritt).

User Interface

Die „Kernmechanik" steht. Nun müssen wir den Spielstatus visualisieren. Dazu brauchen wir ein User Interface.

Zeichnen Sie ein User Interface: Visualisieren Sie die Lebenspunkte und die benötigte Zeit des Spielers. Implementieren Sie dazu die entsprechenden Skripte.

Der Gamemanager

Nun ist das Spiel fast spielbar. Allerdings müssen wir noch das Spiel beenden und wiederholen können. Dazu benötigen wir eine statische Klasse, die über den Spielstatus wacht.

Implementieren Sie eine Managerklasse mit den folgenden Eigenschaften:

- Sobald die Lebenspunkte unter 0 fallen, soll der Spieler an der Startposition respawnen.
- Sobald der Spieler das Ziel erreicht hat (einen Trigger erreicht), soll eine entsprechende Meldung im UI angezeigt werden. Nach fünf Sekunden soll das Spiel wieder losgehen.
- Bomben sollen wieder aktiviert werden.