The Last Super Bowl - Designdokument

Name: Felix Iwertowski

Matrikel-Nr.: 256535

Studiengang: OMB (7. Semester)

Fakultät: Digitale Medien

Fach: Prima

Professor: Prof. Dipl.-Ing. Jirka R. Dell'Oro-Friedl

Semester: Sommersemester 2022

1. Units and Positions

Da das ganze Spiel recht Klötzchen artig aufgebaut ist, sind die Units recht simpel gehalten, erhalten also wenn dann nur eine ,5 hinterm Komma.

Genauer gesagt was die Units der Elemente angeht so sind sowohl die Plattformen als auch der Hero 1 Einheit (X-Achse) breit. In der Höhe gibt es jedoch unterschiede so ist die Plattform hier nur die Hälfte so Hoch wie der Hero ergo Hero = 1 und Plattform = 0,5. Rückblickend hätte man sich hier wahrscheinlich eher für ganze Zahlen entscheiden sollen und dann die Kamera weiter wegziehen sollen, jedoch war das im Moment der Erstellung des Spiels nicht bei den Überlegungen dabei.

Der oder die Footbälle sind etwas größer als der Hero, diese sind in der Breite 2 breit und auch 0,5 Höher als der Hero. Dies sorgt zwar theoretisch für nicht richtige Proportionen, jedoch wurde sich aufgrund der Sichtbarkeit und der Einfachheit des Spiels für diesen Weg entschieden. Gegebenfalls wäre hier auch ein Ansatzpunkt um Schwierigkeitsvarianz in das Spiel zu bekommen, wenn die Footbälle unterschiedliche groß wären beim spawnen.

Die Borders sind nur 0,5 Einheiten breit, dies erklärt sich daraus, dass neben der Farbe eine Unterscheidung zum Hero oder den Plattformen vorhanden sein sollte. Die Höhe von 20 schließt sich aus der möglichen Sprunghöhe bzw. Plattformhöhe, damit man da einfach nie drüber hinweg kann.

Bei der Positionierung wurde sich erstmal Gedanken um die Z-Achse gemacht und sich dann dafür entschieden alle Elemente mit Rigidbody eine Einheit ins positive (z=1) zu setzen zur Abgrenzung aber auch zur Verhinderung von Zusammenstößen und Fehlern.

Die Plattformen wurden platziert nachdem feststand wo die Linke Border hinkommt. Sie wurden also von links nach rechts zur anderen Border gesetzt, jedoch wurde auch hier wieder drauf geachtet, dass maximal die Nachkommastelle 0,5 vorhanden ist. Die Plattformen folgen in der Positionierung eigentlich keinem Muster, sondern sind etwas herausfordernd in der Gegend platziert. Deswegen haben sie auch verschiedene Höhen und auch unterschiedliche Abstände zwischen einander.

Der Hero wird zu Beginn des Spiels auf einer der mittigsten Plattformen platziert bzw. etwas drüber damit direkt am Anfang schon Bewegung ins Spiel kommt und es keinen statischen Eindruck macht von Anfang an.

Ein kleiner Punkt der später nochmal kurz beschrieben wird, ist die Position des Punktes an dem der Hero seinen Tod in der Lava findet. Hier wurde sich mit y=-5 für den ersten Punkt entschieden an dem Hero komplett hinter dem Lava Background verschwindet.

Zu guter Letzt noch die Kamera, diese wurde mit einer Rotation auf der Y-Achse versehen, um den 2D Eindruck in einem 3D Koordinatensystem zu vermitteln.

Sie wird wie später nochmal erwähnt aus der Json Datei geniert. Aufgrund der Rotation, des TranslateZ und der nicht ganz mittigen Platzierung der Elemente wie des Backgrounds wird die Kamera noch in X- und Y- Richtung korrigiert.

Rückblickend wäre hier wahrscheinlich eine ComponentCamera welche an dem Game Graphen hängt sehr viel besser gewesen oder wenn man die Kamera gar nicht wirklich anfässt und auf dem Mittelpunkt des Koordinatensystems belässt. Dann hätte man die grundsätzliche Verschiebung einiger Elemente verhindern können.

2. Hierarchy

Es gibt einen Haupt-Graphen, namens „Game“. An diesen Graphen befinden sich alle weiteren Nodes angehängt. Dieser besitzt an sich erstmal die Light Komponente und jeweils die Audio Komponenten.

Als Knoten führt der Game Graph zum einen den Borders Knoten, welcher wiederum beide Absperrungen links und rechts als Knoten beinhaltet.

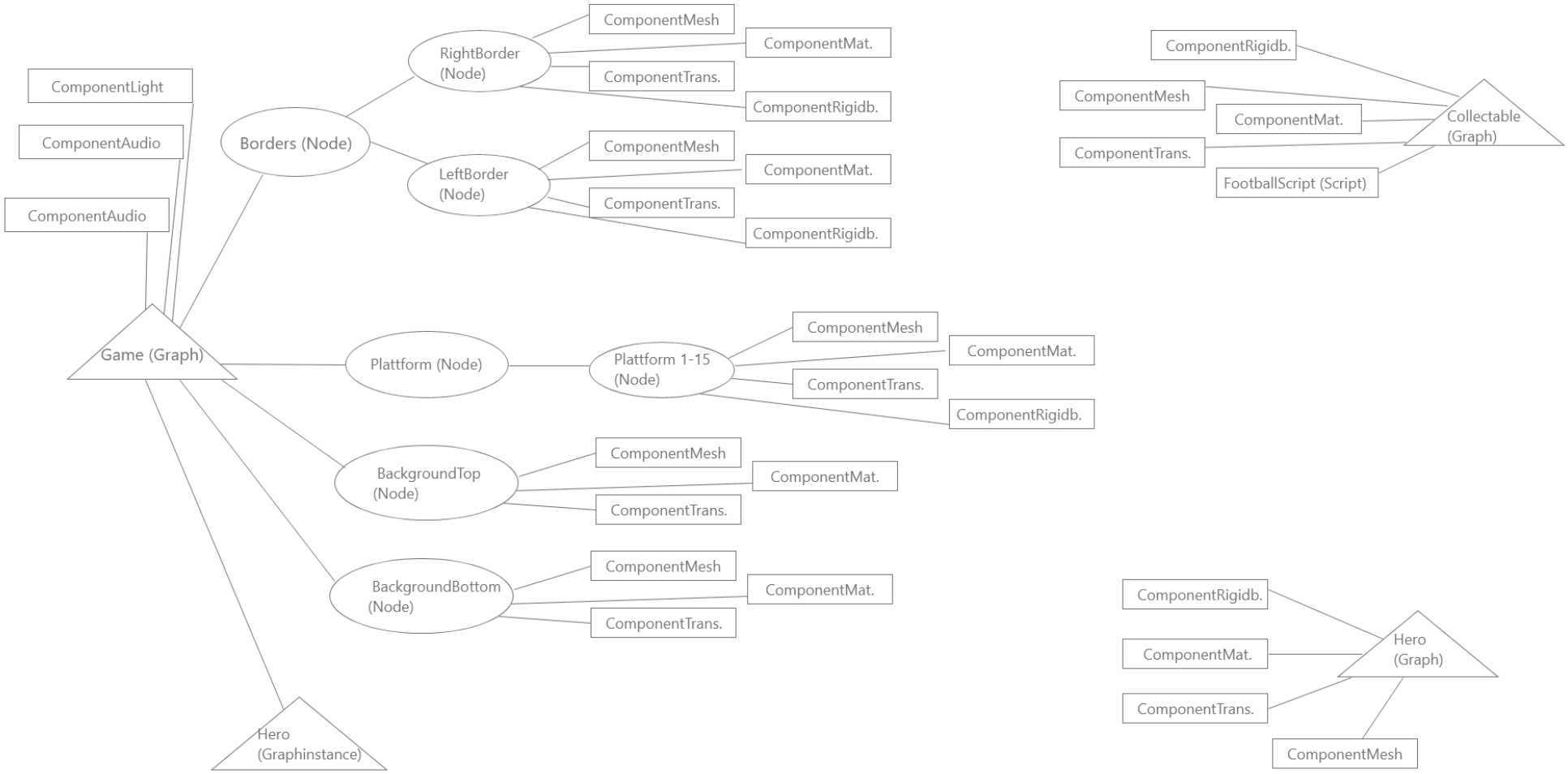
Dann besitzt der Graph den Knoten Plattform, welcher wiederum alle 15 im Spiel vorkommenden Plattformen im als Knoten in sich versammelt.

Sowohl der Borders als auch der Plattform Knoten besitzen dabei keine eigenen Components.

Weiter geht es mit zwei weiteren Knoten am Graphen Game, nämlich BackgroundTop und BackgroundBottom. Diese sind die Hintergrundbilder (Top ist das Stadion und Bottom die Lava) und besitzen dem entsprechend die Components- Mesh, -Material und -Transform jedoch keinen Ridigbody, da diese nicht gehitted werden.

Und zu guter Letzt hängt am Game Graphen noch die Graphen Instanz namens Hero, welche sich aus dem eigenen Graphen Hero bildet.

Außer dem Game Graphen gibt es also noch zwei weitere Graphen so zum einen den Hero Graphen, welcher wie eben erwähnt als Graphen Instanz da ist und auch noch den Collectable Graphen. Dieser wiederum stellt den Football da und wird somit erst in das Spiel geladen, wenn er durch z.B. das Aufsammeln im Code aufgerufen wird.


ComponentLight
ComponentAudio
ComponentAudio
Game (Graph)
Borders (Node)
RightBorder (Node)
ComponentMesh
ComponentMat.
ComponentTrans.
ComponentRigidb.
LeftBorder (Node)
ComponentMesh
ComponentMat.
ComponentTrans.
ComponentRigidb.
Plattform (Node)
Plattform 1-15 (Node)
ComponentMesh
ComponentMat.
ComponentTrans.
ComponentRigidb.
BackgroundTop (Node)
ComponentMesh
ComponentMat.
ComponentTrans.
BackgroundBottom (Node)
ComponentMesh
ComponentMat.
ComponentTrans.
Hero (Graphinstance)
ComponentRigidb.
ComponentMesh
ComponentMat.
ComponentTrans.
Collectable (Graph)
FootballScript (Script)
ComponentRigidb.
ComponentMat.
ComponentTrans.
Hero (Graph)
ComponentMesh

3. Editor

So gut wie alles was man sieht wurde direkt im Editor erstellt und an den vorher besagten Game-Graphen angehängt. Mit ausnahmen des Footballs, welche erst später im Code dann an das Game angehängt wird.

Per Code wird also kaum etwas erstellt, sondern nur mit den nötigen Skripten versorgt. Dies ist darauf zurück zu führen, dass z.B. keine weiteren Plattformen geladen werden müssen, wenn man weiter nach rechts gehen würde, da es sich um einen festen Bereich handelt, im welchem sich der Spieler nur aufhalten kann.

4. Scriptcomponents

Die Script-Komponenten wurden grundsätzlich für den Hero und den Football verwendet um die Berührung abzuprüfen.

Generell ist es dadurch einfach, da man nicht einen gesamten Code schreiben muss sondern sich auf kleinere Teile beziehen kann. So kann man den einzelnen Elementen eigene Scripts mitgeben. Auch sehr hilfreich wenn man es einer Klasse wie in dem Fall dem Football mitgibt und diese es dadurch auch wenn sie neu per Code hinzugefügt wird, einfach anwenden kann.

5. Extend

Alles was im Spiel zusehen ist, sind Unter-Nodes von dem Game Node. Zudem gibt es noch eine GraphInstance also eine eigene Klasse namens Hero, welche wiederum dadurch zu Unterklasse des Game Graphen wird.

Zudem wurde CompontenScript extentend, was sehr hilfreich war um weitere Script-Komponenten im Spiel unterzubringen und zu ergänzen.

6. Sound

Beim Sound wurde sich für zwei unterschiedliche Geräusche bzw. Sounds entschieden. So wird beim Aufsammeln eines Footballs, ein aus vielen Spielen wie z.B. Mario oder ähnlichem bekannten Münzen Sound abgespielt um den Spieler darzustellen, dass er gerade 100 Punkte verdient hat.

Desweiteren gibt es eine Stadion Atmosphäre im Hintergrund welche im Loop permanent spielt um so dem Spieler das Gefühl zu vermitteln, er sei gerade in einem echten Football Stadion.

Die Sounds sind als Komponenten vom Game Graphen im Spiel hinterlegt. Der „Football Einsammeln Sound“ wird aus dem Code aufgerufen und dann

abgespielt, wenn sich der Spieler und der Football berühren. Dies findet in der Main.ts in der Funktion hndPhysics() statt.

Der Background Sound hingegen wird beim Laden der Seite direkt aufgerufen und läuft im Loop ab solange der Spieler spielt.

7. VUI

Das UI ist sehr minimalistisch gehalten, da zum erreichen des Spielprinzips nicht viel angezeigt werden muss. Es gibt einfach nur einen Punktanzeige welche beim Einsammeln von einem Football sich um den Wert 100 erhöht. Damit hat der Spieler einen Wert an dem er sich mit anderen messen kann. Sollte der Spieler abstürzen, ist das Spiel direkt fertig, deshalb ist auch keine Lebensanzeige oder Zeitanzeige von Nöten.

8. Event-System

Das Spiel triggert den Endbildschirm, wenn der Hero in die Lava fällt bzw. unter eine bestimmte Höhe von „-5“ fällt.

Dann wird eine Endscore getriggert bzw. angezeigt und das Spiel wird neugeladen, indem der Spieler das Window neulädt.

9. External Data

Im Ordner „Data“ liegt eine Datei mit dem Namen „testData.json“, diese dient dazu die festen Variablen für die Kamera festzulegen. In der „Main.ts“ werden diese Daten ausgelesen, in der Funktion getJsonData. Die Funktion wird beim Aufbauen per Await Call aufgerufen und im Anschluss werden damit die Translationen und Rotationen für die Kamera Variablen festgelegt.

Hier könnte man noch andere Sachen einbringen wie z.B. könnte man hier die Anzahl Bälle definieren, welche der Spieler einsammeln kann. Somit könnte man das Spiel schwerer oder einfacher machen.

A. Light

Beim Spiel ist Licht und Schatten kein zentrales Game-Element, weswegen alles gleichmäßig beleuchtet werden kann bzw. muss.

Aus diesem Grund wurde sich im während des Aufbaus im Editor nur für ein Ambient-Light entschieden.

B. Physics

Sowohl der Hero als auch die Footballe, welche in das Spiel geladen werden, haben RigidBodys um sich gegenseitig auf Collision zu checken. Die Borders auf den Seiten haben auch feste Bodys um den Spieler nicht an den Seiten rauszulassen.

C. Net

/

D. StateMachine

/

E. Animation

/