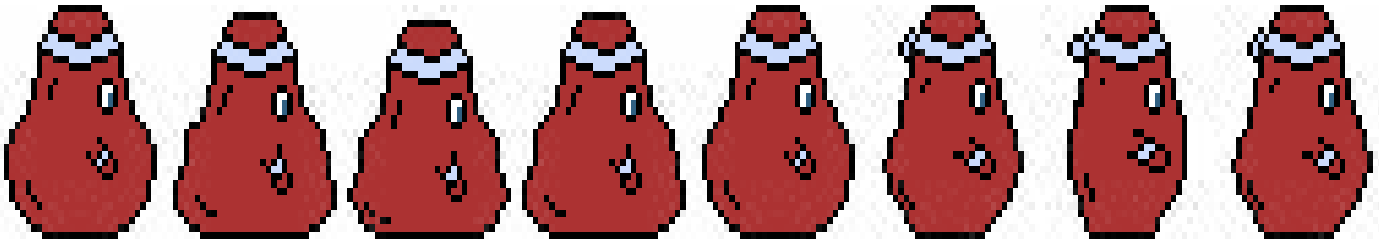
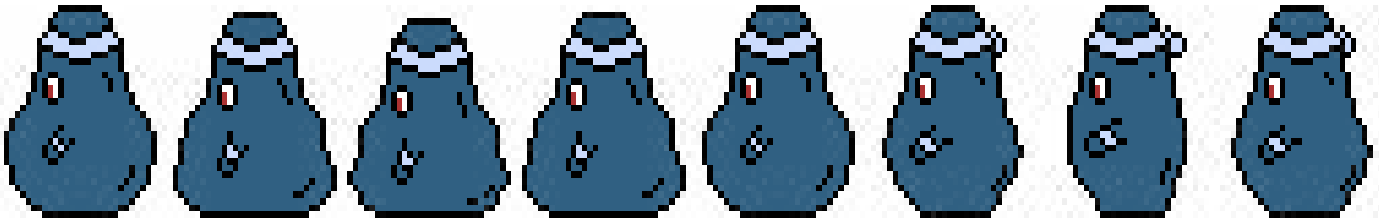
PixelVolley
Design-Dokument

Name:	Florian Malitschenko
Matrikelnummer:	263151
Lehrplansemeter:	6
Studiengang:	MIB
Abgabetermin:	17.02.2023





Prototyping interaktiver Medien-Apps und Games
Wintersemester 2022/2023

Pixel Volley ist ein Arcade-Spiel für zwei Spieler die zeitgleich an einer Tastatur spielen. Dabei ist das spiel recht simpel gehalten. Die Kontrahenten spielen gegeneinander Volleyball. Wer zuerst 10 Punkte hat gewinnt. Alle im Spiel vorkommenden Grafiken und Sound sind eigens hierfür erstellt worden.

Als Vorlage diente das im Jahr 2000 erschienene Spiel Blobby Volley.

1. Units and Positions

Das Koordinatensystem des Spieles ist mittig angesetzt. Der Zentrumspunkt des Sichtfeldes ist dabei der Punkt (0|0|0) im Koordinatensystem. 1 beschreibt einen Meter. Die Blobbs sind somit 0,75 Meter groß.

2. Hierarchy

Die Welt an sich ist der oberste Graph der Hierarchie. An ihm liegen bis auf die Charaktere alle Nodes. Die Charaktere sind in einem extra Graphen, der sich in der Welt befindet, angelegt. Dies ermöglich sie unabhängig von der Welt zu verändern und schneller auf sie zugreifen zu können.

Hierarchisch gestaltet sich der Aufbau wie folgt:

- g_world — erstellt im Editor, oberste Ebene aller Inhalte, enthält LightComponent
  - background — Mesh mit Textur, erstellt im Editor
  - ground — Mesh mit Textur, erstellt im Editor
    - control_blue — Mesh mit Textur, erstellt im Editor
    - control_red — Mesh mit Textur, erstellt im Editor
  - net — Mesh mit Textur und Rigidbody, erstellt im Editor
    - music — MusicComponent, hier angelegt, da Netz immer im Zentrum
  - ball — Mesh mit Textur und Rigidbody, erstellt im Editor
  - wall_right — Mesh mit Textur und Rigidbody, erstellt im Editor
    - above_wall — Mesh mit Rigidbody, erstellt im Editor
  - wall_left — Mesh mit Textur und Rigidbody, erstellt im Editor
    - above_wall — Mesh mit Rigidbody, erstellt im Editor
  - rigid_left_point — Rigidbody unterhalt des roten Blobbs
  - rigid_right_point — Rigidbody unterhalt des blauen Blobbs
  - g_character — Graph für den gezielten Zugriff auf die Blobbs
    - character_red — Rigidbody, erstellt im Editor
      - rigid_lower — Rigidbody, erstellt im Editor
    - character_blue — Rigidbody, erstellt im Editor
      - rigid_lower — Rigidbody, erstellt im Editor
  - sun — Mesh mit Textur und LightComponent, erstellt im Editor

Zusatz: Jeder Rigidbody besitzt auch eine Transform Kompoonente.

3. Editor

Es lässt sich grundsätzlich sagen, dass bis auf die Sprites der Blobbs, alle Inhalte im Editor angelegt wurden. Dies ermöglichte mir eine visuelle Grundvorstellung des Aufbaus. Die einzelnen Nodes und Components werdenn zur Laufzeit des Spieles im Code manipuliert.

4. Scriptcomponents

Die Sonne, die im Spiel rotiert, wird per Scriptcomponent gesteuert. Dazu wurde das SunRotationScript.ts angelegt. Für die Umsetzung meiner Idee war dies eigentlich nicht zwingend erforderlich, daher habe ich sie nur als potentielle Lichtquelle eingefügt.

5. Extend

Die Spielercharaktere (Blobbs) werden in eigenen Klassen (characterBlue.ts und characterRed.ts) instanziiert. Hier werden die Blobbs und Ihre Animationen erstellt. Das ermöglicht, wenn gewünscht schnell neue Blobbs anzulegen und diese in die Spielszene einzufügen.

6. Sound

Es gibt einen Hintergrund-Soundtrack und einen „Ball-Hit" Sound, der abgespielt wird, sobald ein Blobb den Volleyball mit dem Kopf spielt. Dieser dient der auditiven Unterstützung des Spielers und wird in der Funktion playedSound der Main. ts angelegt.

7. Interface

Das Interface ist aus der Szene selbst ausgelagert und beinhaltet nur den Punktestand. Warum ausgelagert? Ich war der Meinung, dass es in der Szene zu sehr ablenken würde. Gesteuert wird dieses Interface über die manipulation der HTML-Inhalte per getElementByID und das auslesen des StandingsCounter.ts.

8. Event-System

Es gibt einen hinterlegten Event-Listener, der Kollisionen der Rigidbodys abfängt und die Events weiterleitet. Dabei ist der „Ball-Hit" Sound über dieses gesteuert, sowie der Punktecounter. In diesem Fall war der Eventtrigger nicht nur hilfreich, sondern auch erforderlich. Da der Ball das Objekt ist, mit dem alle relevanten Events getriggert werden, wurde das Event-System auch hier angelegt.

9. ExternalData

Im Ordner Script/Source liegt eine JSON-Datei mit dem Namen config.json. In dieser können die Richtwerte angepasst werden, ohne den gesamten Code durchstöbern zu müssen. In configRec.json findet man die von mit empfohlenen Werten.

A. Light

Die rotierende Sonne wurde als Lichtquelle verwendet.
Da der Fokus des Spiels auf dem Volleyballfeld liegen soll und nicht auf den Lichteffekten im Hintergrund wurden die umliegenden Texturen als „ShaderLit" angelegt, was dazu führt, dass das Ambient Light der Sonne keinen Einfluss auf die Lichtstimmung des Spieles nimmt.

B. Physics

Alle Meshes besitzen Rigidbodies, mit denen kollidiert werden kann. Zudem werden die Blobbs bewegt, indem ihnen eine Force im4 movement angehängt wird. Das Verhalten des Balles wird zudem durch seine Masse und die auf ihn einwirkenden Impulse durch die Blobbs sowie die Schwerkraft gesteuert.

C. Net

Eine Netzwerkkomponente ist nicht implementiert, da das Spiel als lokaler Multiplayer an einer Tastatur ausgelegt ist. Es ist jedoch möglich, das Spiel um diese Komponente zu erweitern.

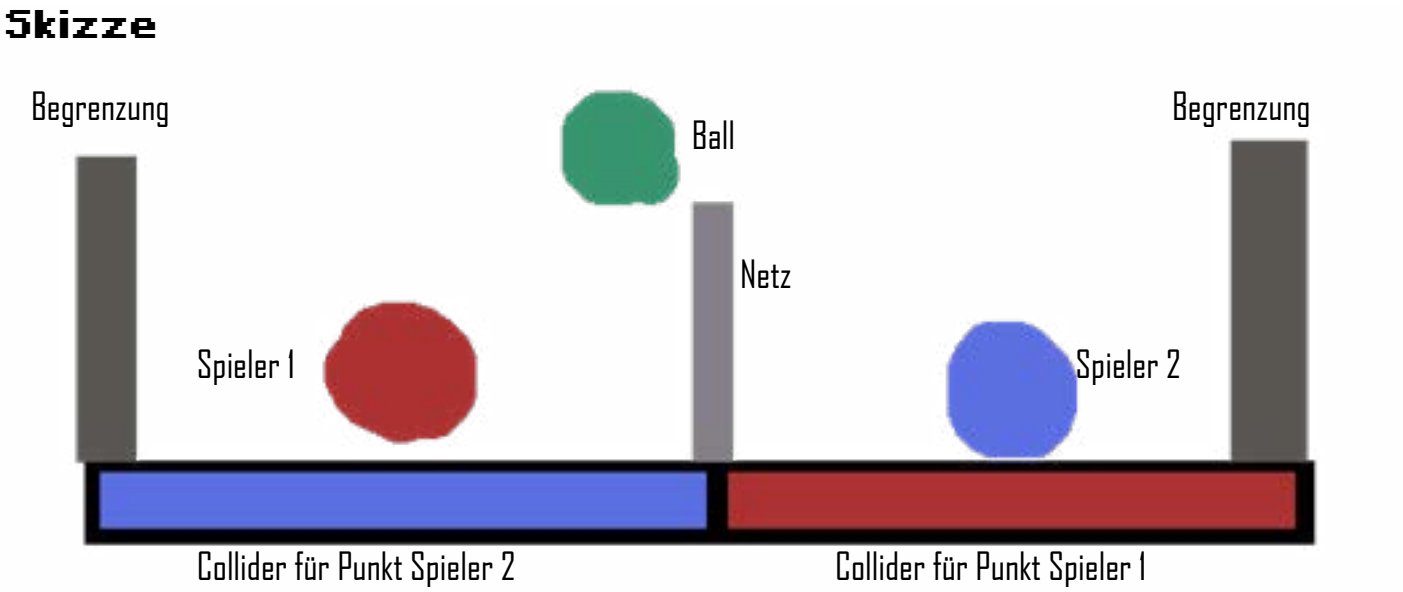
D. StateMachines

State Machines wurden nicht implementiert.

E. Animation

Die Animationen der Blobbs werden per Spritenodes generiert. Dies geschieht in der jeweiligen Klasse der Blobbs (characterBlue.ts und characterRed.ts), um eine schnelle Erweiterung zu ermöglichen.

Skizze



Sprites und Texturen

Alle im Spiel vorkommenden Grafiken wurden eigens hierfür in Aseprite erstellt.




Dateien und ihre Inhalte

Main.ts
- start
- played Sound
- checkEnd
- update

StandingsCounter.ts
- counterRed
- counterBlue

SunRotationScript.ts
- rotate

characterBlue.ts und characterRed.ts
- initBlob
- valueAnimation
- loadTextureFromSpriteSheet
- movement
- buildAnimation
- createNewSpriteNode

config.json
- jumpImpulse
- backwardForceBlue
- backwardForceRed
- ballMass
- maxScore
- forwardForceBlue
- forwardForceRed
- ballGravity