

# Dodgeball Simulator

Jan Kompatscher (01527584)

Doris Rhomberg (01649642)

## Implementierung

Von der **Main** aus werden alle weiteren Klassen aufgerufen.

Im **Window** wird das Fenster erstellt.

In der Klasse **Settings** werden alle Interaktionen implementiert. Die intuitive Steuerung beinhaltet Gehen, Laufen, Herumschauen und Werfen des Balles. Der Ball wird in Richtung eines Punktes etwas über der Bildschirmmitte geworfen, damit die Gegner nicht vom Spieler selbst verdeckt werden, wenn auf sie gezielt wird. Außerdem wird in dieser Klasse die Klasse **INIReader**, welche die gesetzten Parameter aus dem **settings.ini** herausliest. Hier muss erwähnt werden, dass unser Spiel funktioniert, wenn die Breite und Höhe des Screens unter der Bildschirmauflösung liegt. Auf den Vislab PCs funktioniert es bis zu der Größe 1920x1020.

In den Klassen **Model** und **Mesh** werden die Modelle mittels Assimp geladen.

Die Klasse **Ball** erstellt einen Ball.

Unsere **Camera** ist eine 3rd Person Kamera. Sie rotiert um den Charakter, der sich jedoch nicht mitdreht, und kann nicht höher als 50 Grad und niedriger als -40 Grad bewegen.

**CollisionCallback** meldet sich, wenn eine Kollision passiert und ruft dann `Enemy/PlayerCharacter.hit()` auf.

**Enemy** implementiert die Gegner, die sich random bewegen. **PlayerCharacter** implementiert den eigenen Spieler.

In der Klasse **Physics** wird die Physik mittels physx realisiert.. Also die 6 Wände der Turnhalle, der Ball, das Werfen des Balles und die Kollision von Ball mit entweder dem Spieler oder den Enemys.

**Shader** erzeugt Shader.

**Text\_Renderer** erstellt und rendert Text mit der FreeType Library.

In der Klasse **Particle** werden Konfettis erzeugt. Sie erscheinen, wenn der Spieler das Spiel gewonnen hat.

## Features of the Game

- Vom Start Game Screen wird mit "ENTER" das Spiel gestartet.
- Bewegen mit WASD, laufen durch gleichzeitiges Drücken von shift.
- Der Ball geht automatisch an den Charakter über, in dessen Feld er zur Ruhe kommt.
- Werfen des Balles durch Drücken der linken Maustaste.
- Zurücksetzen des Spiels durch Drücken von „R“.
- Aktivieren/deaktivieren des HUDs durch Drücken von „H“.
- Anzeige der Leben des Spielers mit Hilfe von Herzen im linken oberen Eck.
- Anzeige ob der Spieler den Ball hat durch Ball-zeichen im rechten oberen Eck.
- Anzeige des Spielstandes in der oberen Mitte des Bildschirms.
- Spielende, wenn einer der Teams 3 Punkte erzielt hat.
- GameOver und GameLost Screen.

## Illumination

Wir verwenden Lightmaps. In der Szene selbst haben wir aber kein Licht implementiert. Die selbe Lightmap wird auf alle statischen Flächen, also alle Flächen der Turnhalle, angewendet.

## Effects

Mittels **Bloom/Glow** leuchtet unser Loch in der linken Wand der Turnhalle. Es soll wie ein Fenster wirken. Bei der Implementierung haben wir uns sehr stark an das Tutorial von <https://learnopengl.com/Advanced-Lighting/Bloom> orientiert. Wir verwenden jedoch keinen Lightshader, da wir keine Lichtquellen in unserem Spiel haben. Dieser Effekt ist im Spielscreen (Screen 2) zu sehen.

Auf der rechten Wand der Turnhalle haben wir eine **Videotextur** angebracht. Sie soll das Publikum darstellen. Die roten Figuren der Textur bewegen sich wenn der Spieler in Führung liegt und die blauen, wenn die Gegner in Führung liegen. Die Videotextur besteht aus 16 verschiedenen Bildern, von denen immer das nächste nach einer deltaTime von einigen Sekundenbruchteilen geladen gezeichnet wird. Dieser Effekt ist im Spielscreen (Screen 2) zu sehen.

Mittels **Lightmaps using separate Textures** ist die Turnhalle in den Ecken dunkler und wirkt reeller. Dieser Effekt ist immer zu sehen.

Wenn der Spieler gewonnen hat, fliegen Konfettis auf den Spieler. Sie wurden als **Particle** implementiert. Wir haben das Tutorial von <http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/> für die Particles befolgt. Dieser Effekt ist im Gewonnen-Screen (Screen 4) zu sehen.

## Texturen

Die Texturen sind an die mit Assimp eingebundenen Modelle, die mit Blender erstellt wurden, angefügt. Die Textur wird dann im Fragment Shader durch entsprechende Kolorierung der entsprechenden Fragmente auf die Modelle aufgetragen.

Modell Spieler: <https://free3d.com/3d-model/little-boy-36169.html>

Texturen Turnhalle: <https://www.cg.tuwien.ac.at/courses/Textures/>

Textur Ball: <https://www.cgtrader.com/free-3d-models/sports/toy/soviet-toy-ball>

Die Videotexturen wurden selber mit GIMP gezeichnet und als JPEGs gespeichert.

Die Lightmap wurde im Blender erstellt.

## Zusätzliche Bibliotheken

Assimp: <https://github.com/assimp/assimp/releases/tag/v4.1.0/>

Freetype: <https://sourceforge.net/projects/freetype/files/freetype2/>

GLEW: <http://glew.sourceforge.net/>

GLFW: <https://www.glfw.org/download.html>

GLM: <https://glm.g-truc.net/0.9.9/index.html>

PHYSX: <https://developer.nvidia.com/physx-sdk>