# Bomberman – Prototyp

Designdokument
PRIMA Wise21/22
Prof. Jirka Dell´Oro-Friedl

vorgelegt von
**Attinger, Kevin**
**Mtr.Nr:** 259010

# Inhaltsverzeichnis

## Beschreibung des Prototyps:

Ich denke dieser Prototyp bedarf keiner großen Beschreibung. Jeder kennt vermutlich den hier behandelten Klon – Bomberman.

Man steuert eine Spielfigur durch ein Spielfeld voller Blöcke, dabei gibt es Wände, unzerstörbare Blöcke und zerstörbare Blöcke. Des Weiteren lauern an jeder Ecke des Spielfelds NPC's.

Bei diesem Klon handelt es sich lediglich um einen Prototypen – heißt in jedem Funktionsumfang klein gehalten, viele Funktionalitäten wurden nur angedeutet um einen PoC (proof of concept) zu bieten. Mehr dazu im Punkt ,,Zukunftsaussichten des Prototyps‘‘ innerhalb dieses Designdokuments.

Programmiert wurde dieser Prototyp mit Typescript und der Engine FUDGE innerhalb des Kurses Prototyping interaktiver Medien – Apps und Games (PRIMA).

## Funktionsumfang des Prototyps

Der Prototyp generiert die Map auf Grundlage verschiedener Werte innerhalb einer Mapsize.JSON datei. Dabei wurden verschiedene Auswahlmöglichkeiten verschiedenen Werten innerhalb der Mapsize.JSON zugewiesen. Zu beachten ist hierbei da das generieren der Wände, zerstörbaren Blöcken und unzerstörbaren Blöcken ebenfalls auf dieser Auswahl basiert, wodurch es zu empfehlen ist das wenn an der Mapsize.JSON ,,gespielt‘‘ wird man immer in 3er Schritten arbeitet um gewährleisten zu können das die Map ihre Symmetrie beibehält.

Die Kernfunktionalität des Spiels bildet das Zerstören von Blöcken ab. Dabei sei zu beachten das nur manche Blöcke auf dem Spielfeld zerstörbar sind. Diesen Aspekt der Funktionalität bildet der Prototyp in seiner Gesamtheit ab.

Ein weiteren zentralen Aspekt des Prototypens stellen die NPCs dar. Dabei wurde innerhalb der Statemachine mit Physics Collision gearbeitet. Die KI wurde aufgrund der zeitlichen knappheit relativ simpel gehalten, ändern aber entsprechend ihrer collision ihr Verhalten. Als beispiel:
Wenn collision destroyable Block -> transit placeBomb.
Wenn collision Wall -> transit turn ; transit Walk.
Näheres dazu natürlich innerhalb des Codes zu finden.

## Problematiken des Prototyps

Zentrale Probleme des Prototypens sind zum einen Performance. Ab einer gewissen Mapgröße fängt die Applikation an zu ,,stottern‘‘ bzw. zu laggen. Mutmaßungen

meinerseits sind die Berechnungszeiten der Physics da eben jeder Block einzeln erstellt und jeweils ein Rigidbody hinzugefügt wird. (Außer bei der Base, die wird im FUDGE-Editor erstellt und dann bei dem Start des Games gescaled).

Ein weiteres Problem sind die unausgereiften Gamemechaniken, im Original gibt es einen Timer, nach Ablauf der Zeit gibt es einen ,,Todesmodus‘‘, ebenfalls existieren dort auch Items welches das Spiel ,,attraktiver‘‘ für den Spieler machen usw. – mein Spiel wird bereits nach dem zweiten Spielen relativ eintönig, da meine Schwerpunkte auf der Funktionalität der Generierung der Map, zuverlässigen Bomb Collision und NPC’s lag.

## Zukunftsaussichten des Prototyps

Die Kernfunktionalität der angesprochenen für mich wichtigen Elemente Mapgenerierung, Collisionen und NPC’s sind meiner Meinung nach zuverlässig implementiert. Bezüglich der Mapgenerierung muss unbedingt geklärt werden inwieweit die Perfomance ab einer Größe optimiert werden kann, bzw. ob optimiert werden kann und wenn ja – wie?

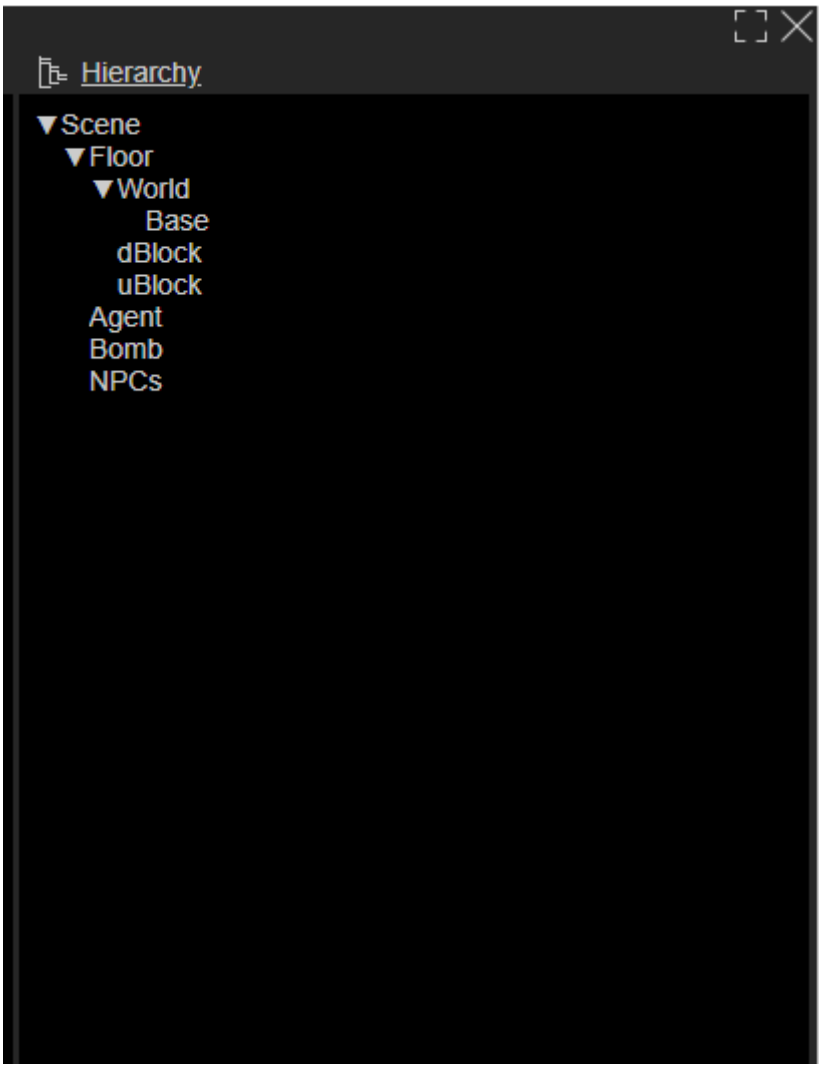
Es gilt nun den Prototypen vor allem in Sachen neue Gamemechaniken, Item Implementierung, Texturen, Optik Anpassungen bzgl. Agent und NPC’s zu verbessern um wie bereits vorher angesprochenen aus diesem Proof of Concept ein richtiges Spiel zu machen.

Um gewährleisten zu können das dieser Prototyp in der Zukunft weiter reift wäre allerdings meine Handlungsempfehlung ein Code – review. Da ich als Architekt dieses Prototyps erst ein Semester Erfahrung mit der Engine FUDGE sammeln konnte gibt es mit ziemlicher Sicherheit an einigen Stellen optimierungspotenziale – Logik Verbesserungen und auch Ressourcen Einsparungen.

# Anforderungen

| Nr | Criterion | Explanation |
|---|---|---|
| 0 | Units and Positions | Every block has a width and height of 1; the Agent and the NPCs are half of that aswell as the bombs. Flames are 1 in every direction aswell. |
| 1 | Hierarchy | A screenshot of the Hierarchy can be found in the Design Document (linked above). |
| 2 | Editor | I used the Editor to create only one 1x1 Block which builds my base. Everything else is added via Code because the Mapsize can be modified by a user which i will explain on 8) External data. |
| 3 | Scriptcomponents | My agent has a Scriptcomponent which controls things like movement or movement speed in gernal. |
| 4 | Extend | Agent, Bomb, NPC, Flames inherit from f.Node to provide the functionality of adding components and graph placement |
| 5 | Sound | My game has different Hit markers one for the Agent, one for the NPCs. Also if a bomb explodes it will return a Sound. |
| 6 | VUI | The UI consits of a „Health - Bar'' for the Agent. It has an initial value of 2, if the Agent got hit 2 times by a Bomb and the Gamestate is equal to zero there will be a Gamerover screen popping up. |
| 7 | Event-System | The eventsystem is used when flames from a bomb are created. It will fire a explode event which contains the flame position (i modified the event in the DataEvent.ts a bit), where i then can check if it has the same positons as a „Destroyable Block'' position, if true the block will remove itself. |
| 8 | External Data | I implemented a Mapsize.JSON in this project. At the start of the game the user can selected between three different mapsizes. Depending on the user selection the correct size of the map will be fetched and will then do the following: scale the initiale block created in the fudge editor to the size, create walls, and destroyable blocks all depending on this selection. |
| 9 | Light | A ambient light was placed on the floor node to provide a base luminance and aswell as a directional light. |
| A | Physics | Rigidbody components: Agent, NPCs, Walls, Destroyable Blocks, Base. Forces on Agent. Collision Eventhandling on NPC's with Destroyable Blocks, Walls and Base. (also mentioned in the following - Statemachine |
| C | State Machines | My Statemachine controls the behaviour of my NPC by checking if they collide, and especially with who they collide, depending on that the statemachine transits into a different state and changes the behaviour of the NPC. |

## Hierarchy


Hierarchy
▼Scene
▼Floor
▼World
Base
dBlock
uBlock
Agent
Bomb
NPCs