Bomberman Game Design Document

Jerry Hu, 6229077
Maaike Koninx, 6303919

November 12, 2017

1 Game Description

Bomberman is an action strategy game. The goal of the game is for the player to eliminate enemies by placing down bombs. Bombs will explode in several directions after an amount of time. The game is played on a 15 x 13 or 17 x 15 grid.

1.1 Player

The player will control a character which can walk around the map and place bombs. The player can move the character by using WASD or arrows of a keyboard. By pressing [,] he/she can plant a bomb. A bomb can only be planted after a given interval.

1.2 Map

The map is a grid that contains Stone- or Metal Blocks. Metal Blocks can not be destroyed by a bomb. The Metal Blocks are placed every two blocks on the field. The destructible blocks are randomly placed on the map. The player and his/her enemies will first have to destroy several blocks to reach each other. Destructible blocks can also drop power-ups which enhances the abilities of a player.

1.3 Enemies

Enemies in the game will move around the map randomly. The enemy can be defeated by hitting it with an explosion from a bomb. When a player intersects with an enemy or explosion he/she will lose. The enemies can not plant bombs.

1.4 Power-ups

Blocks can drop several power-ups:

- Speed boost, enhances the movement speed of the player by 2

- Bomb Speed, lowers the interval a player can place a bomb
- Extra Bomb, increases the amount of bombs that can be placed.

1.5 Animations

Several objects will be animated during gameplay:

- Player movement
- Enemy movement
- Bomb explosion

1.6 Score

The score is calculated by the time it takes for the player to defeat all enemies. If the player does not finish the level, the score will be calculated by the elapsed time. The highscore is written and read from a file.

1.7 Randomness

Randomness is used in several ways. The enemies will move randomly across the grid and PowerUps are randomly dropped when Stone Blocks are destroyed.

1.8 Controls

- "ESC" Pause the game
- "," Plant a bomb
- Arrow keys [UP,DOWN,LEFT,RIGHT], move in a direction
- "SPACE", Used inside menu's for quitting the game or continuing.

2 Data types and Typeclasses

The game will have several data types and typeclasses.

2.1 Data types

The following data types are described by their names and attributes

2.1.1 Player

The Player type represents characters that can move and interact in the game. The player controls a Player, enemies are also players but are AI controlled.

```
data Player = Player {
          name             :: String,
          health           :: Health,
          playerPosition   :: Pos,
          velocity         :: Vel,
          playerDirection  :: Direction,
          goal             :: Pos,
          state            :: PlayerState,
          sprite           :: Picture,
          explosionSpeed   :: Int,
          timeTillNewBomb  :: [Int]
}deriving(Eq)

Instance of Movable, HasArea, Positioned and Renderizable
```

2.1.2 GameState

The GameState type keeps the core state of the game. The GameState is updated each frame and also displayed in the view.

```
data GameState = GameState {
        player        :: Player,
        grid          :: Grid,
        currentState  :: CurrentState,
        gen           :: StdGen,
        keyState      :: KeyState,
        enemies       :: Enemies,
        bombs         :: Bombs,
        explosions    :: [Explosion],
        powerUps      :: [PowerUp],
        elapsedTime   :: Float
}

Instance of Renderizable
```

2.1.3 Bomb

```
data Bomb = Bomb {
        bombPosition       :: Pos,
        timeTillExplosion  :: Float,
```

```
        spriteBomb          :: Picture
}
```

2.1.4 PowerUp

```
data PowerUp = PowerUp {
        powerUpPosition :: Pos,
        powerUpType     :: PowerUpType
}
data PowerUpType = SpeedBoost | ExtraBomb | FasterBomb

Instance of Renderizable, Positioned, HasArea
```

2.1.5 GameObject

```
data GameObject = MetalBlock | StoneBlock
                  deriving(Show, Ord, Eq)
```

2.1.6 Field

A Field contains its position and a specific GameObject.

```
data Field = Field {
    fieldPosition :: Pos,
    gameObject :: GameObject
} deriving (Eq)
```

2.2 Typeclasses

The following typeclasses are used to generalize concepts in the game.

- Positioned, used for types that have an x and y position in the game
- HasArea, used for types that have a boundary box which is needed for collision detection
- Movable, used for types that can move across the grid.
- Renderizable, used for types that have a Sprite or shape that can be displayed on the View.