



Dokumentation von Max Schwarz

Teamname	SoulThieves
Rolle(n)	Software Developer

Aufgaben:

- Mapper
- Bomben Kicken
- Bomben Werfen
- Aus der Welt fallen (Spieler/Client)
- Pfeile
- Bombmortar
- Eis

Gliederung:


1. Projektdokumentation
 - a. Aufgaben
 - b. Protokolle
2. Mapper
3. Bomben Bewegung
 - a. Rutschen
 - b. Fliegen
 - c. Interaktion mit Tiles
 - d. Interaktion mit Spielern

1 Projektdokumentation

a. Aufgaben

Untitled Database

b. Protokolle

 Untitled Database

2 Mapper

Im Spiel gibt es mehrere Maps die aus dem Spiel Granatier von KDE (<https://apps.kde.org/de/granatier/>) übernommen wurden. Diese liegen als XML Datei vor.

```
<?xml version="1.0"?>
<Arena arenaFileVersion="1" rowCount="13" colCount="17">
  <Row>===== </Row>
```

```

<Row>=p_d+=++++d_p=</Row>
<Row>=++++++++=</Row>
<Row>=++++d++++l+=</Row>
<Row>=+++++=++++=</Row>
<Row> =+=xx+= </Row>
<Row> -----xox----- </Row>
<Row> =+=xx+= </Row>
<Row>=+++++=++++=</Row>
<Row>=++++u++++l+=</Row>
<Row>=+++++=</Row>
<Row>=p_u+=++++u_p=</Row>
<Row>===== </Row>
</Arena>

```

Im Spiel gibt es das Script Map.gd in der Node "TileMap" in der Scene "World". Dieses ist verantwortlich für die Verarbeitung der Map XML Dateien.

Es gibt eine Mappingtabelle die Zeichen aus der XML Dateien in Tilennamen übersetzt:

```

const MAP_CHAR_TO_NAME = {
  '=': 'arena_greenwall',
  ' ': ' ',
  '_': 'arena_ground',
  '+': 'arena_wall',
  '-': 'arena_ice',
  'o': 'arena_bomb_mortar',
  'u': 'arena_arrow_up',
  'r': 'arena_arrow_right',
  'd': 'arena_arrow_down',
  'l': 'arena_arrow_left',
  'm': 'arena_mine',
}

```

Die XML Datei wird durchlaufen und für jedes Zeichen die Funktion `set_cell_from_char` mit dem Zeichen und der Koordinaten an der Stelle. Diese setzt dann mit Hilfe der Mappingtabelle die Stelle in der "World" auf das richtige Tile

```

func set_cell_from_char(x,y,val):
  if(val == 'x'):
    var rng = RandomNumberGenerator.new()
    rng.randomize()
    if(rng.randf_range(0,1)>0.5):
      val='_'
    else:
      val='+'
  if(val == 'p'):
    val='_'
    var spawn = Position2D.new()
    spawn.position = map_to_world(Vector2(x,y))
    spawn.name = "PlayerSpawn" + str(spawn_count)
    spawn_count += 1
    add_child(spawn)
  set_cell(x,y,tile_set.find_tile_by_name(MAP_CHAR_TO_NAME[val]))

```

3 Bomben Bewegung

Alle Features die zum Bewegen der Bombe Notwendig sind liegen in dem Script Bomb.gd in der Scene "Bomb". Die Bomb kann exklusiv entweder sich nicht bewegen, zu einem Feld fliegen oder in eine Richtung rutschen bis sie mit etwas kollidiert.

a. Rutschen

Um eine Rutschbewegung zu starten wird die Function `move()` mit einer Richtung als String aufgerufen. Erst wird die bisherige Bewegungsrichtung in einer Temporärenvariable Gespeichert. Der Richtungs String wird dann in einen Richtungsvector übersetzt und in `slide_dir` gespeichert.

Anschließend wird geprüft ob das nächste Feld frei ist.

Wenn es nicht frei ist wird `slide_dir` auf den Nullvektor gesetzt.

Wenn die Bewegungsrichtung sich seit dem letzten Aufruf geändert hat, wird die Bombenposition auf die Mitte der aktuellen Zelle gesetzt.

```

func move(dir):
    var old_slide_dir=slide_dir
    var next_field_free
    match dir:
        "up":
            slide_dir=Vector2.UP
        "down":
            slide_dir=Vector2.DOWN
        "left":
            slide_dir=Vector2.LEFT
        "right":
            slide_dir=Vector2.RIGHT

    next_field_free = !test_move(Transform2D(Vector2(self.scale.x,0),Vector2(-0,self.scale.y)),get_center_coords_from_cell_in_world_coord

    if slide_dir!=old_slide_dir:
        self.position=get_center_coords_from_cell_in_world_coords()

    if not next_field_free:
        slide_dir=Vector2.ZERO

```

Die eigentliche Bewegung wird in der Funktion `_physics_process(delta)` ausgeführt.

```

if slide_dir!= Vector2.ZERO and not exploding:
    var field_free = !test_move(Transform2D(Vector2(self.scale.x,0),Vector2(-0,self.scale.y)),get_center_coords_from_cell_in_world_coord
    var collision_info
    if field_free:
        collision_info = move_and_collide(delta*slide_dir*200)
    if collision_info or !field_free:
        slide_dir=Vector2(0,0)
        self.position = get_center_coords_from_cell_in_world_coords()

```

Wenn die Bewegungsrichtung `slide_dir` nicht der Nullvektor ist und die Bombe nicht gerade explodiert dann wird geprüft ob das nächste Feld in Bewegungsrichtung frei ist.

Wenn es frei ist wird die Bomb mit `move_and_collide(delta*slide_dir*200)` bewegt (delta ist die Zeit die Seit dem Letzen aufruf von `_physics_progress` vergangen ist) und in der variable `collision_info` gespeichert ob die Bombe dabei mit etwas kollidiert.

Sollte das nächste Feld nicht frei sein dann wird der Bewegungsrichtungsvektor auf den Nullvektor gesetzt und die Bombe auf die Mitte der aktuellen Zelle gesetzt.

b. Fliegen

Um eine Flugbewegung zu starten wird die Function `throw()` mit einem Zeilvector aufgerufen

```

func throw(destination):
    if move["dest"] == null:
        $ExplosionTimer.set_paused(true)
        get_node("BombAnim/AnimationPlayer").stop(true)
        var tilemap = get_parent().get_parent().get_node("TileMap")
        move["dest"]=tilemap.map_to_world(destination)+Vector2(20,20)+tilemap.position
        move["length"]=self.position.distance_to(move["dest"])
        move["dir"]=self.position.direction_to(move["dest"])
        move["progress"]=move["length"]
        slide_dir=Vector2.ZERO
        $CollisionShape2D.set_deferred("disabled", true)

```

Alle zugehörigen Variablen werden in der Map "move" gespeichert.

Zu erst wird geprüft ob die Bombe sich schon im Flug befindet. Wenn das nicht der Fall ist dann wird zuerst die Bombenanimation und der Bombentimer gestoppt.

In `move["dest"]` werden die Zeilkoordinaten gespeichert.

In `move["length"]` wird die Länge des Vectors zwischen der aktuellen Position und dem Zeil gespeichert.

In `move["dir"]` wird der Richtungsvektor Richtung Ziel gespeichert.

In `move["progress"]` wird initial auch die Länge des Zeilvektors gespeichert

Der `slide_dir` Vektor wird auf den Nullvektor gesetzt um bisherige Rutschbewegungen zu beenden.

Es wird außerdem alle Kollision deaktiviert.

Die eigentliche Bewegung wird in der Funktion `_physics_process(delta)` ausgeführt.

```
elif move["dest"] != null:
    if move["progress"] <= 0:
        self.scale = Vector2(0.2, 0.2)
        self.position = move["dest"]
        var players_contained
        var intersections = $PlayerIntersection.get_overlapping_bodies()
        for intersection in intersections:
            if intersection.is_in_group("Players"):
                players_contained = true
            if intersection.is_in_group("LocalPlayers"):
                intersection.in_bomb = self
        if not players_contained:
            $CollisionShape2D.set_deferred("disabled", false)

        move["dest"] = null
        move["length"] = null
        move["dir"] = null
        $ExplosionTimer.set_paused(false)
        get_node("BombAnim/AnimationPlayer").play()
    else:
        if move["progress"] > move["length"] / 2:
            var portion_traveled = (move["progress"] - (move["length"] / 2)) / (move["length"] / 2)
            self.scale = Vector2(0.3 - 0.1 * portion_traveled, 0.3 - 0.1 * portion_traveled)
        else:
            var portion_traveled = (move["progress"]) / (move["length"] / 2)
            self.scale = Vector2(0.2 + 0.1 * portion_traveled, 0.2 + 0.1 * portion_traveled)
        var move_tmp = (move["dir"] * 5)
        self.position = self.position + move_tmp
        move["progress"] = move["progress"] - move_tmp.length()
```

in `move["progress"]` wird die Verbleibende Flugstrecke gespeichert. Wenn diese ≤ 0 wird die Bewegung beendet und abhängig ob die Bombe sich gerade in einem Spieler befindet die Kollision wieder aktiviert.

Wenn `move["progress"]` nicht ≤ 0 ist wird abhängig davon ob die Bombe sich in der ersten oder zweiten Hälfte der Flugstrecke ist die Größe der Bombe angepasst.

Außerdem wird die Bombe um das 5 fache von `move["dir"]` bewegt.

`move["progress"]` wird außerdem um die Länge der oberhalb genannten Strecke reduziert.

c. Interaktion mit Tiles

Die Funktionen `throw()` und `move()` der Bombe können aufgerufen werden wenn die Bombe sich auf bestimmten Feldern befindet. Das findet in der `_process()` function statt.

```
func _process(_delta):
    if not exploding:
        var celltype = get_celltype_from_coords()
        if moving_bomb_is_more_than_half_on_cell():
            if not move["dest"]:
                match celltype:
                    "arena_arrow_up", "arena_arrow_down", "arena_arrow_right", "arena_arrow_left":
                        move(celltype.right(12))
                    "arena_bomb_mortar":
                        self.position = get_center_coords_from_cell_in_world_coords()
                        var tilemap = get_node("../TileMap")
                        randomize()
                        var dest_x = randi() % int(tilemap.columns)
                        var dest_y = randi() % int(tilemap.rows)
                        throw(Vector2(dest_x, dest_y))
            else:
                if moving_bomb_is_more_than_half_on_cell():
                    player.stats.layable_bombs += 1
                    queue_free()
        else:
            slide_dir = Vector2(0, 0)
            self.position = get_center_coords_from_cell_in_world_coords()
```

Wenn die Bombe nicht gerade Explodiert und nicht fliegt wird abhängig vom Celltype für

- `arena_bomb_mortar` throw mit einer Zufälligen Position auf dem Feld aufgerufen.

- arena_arrow_up oder arena_arrow_down oder arena_arrow_right oder arena_arrow_left mit dem Richtungssubstring aus dem Feldnamen aufgerufen z.B. "up" oder "right"

d. Interaktion mit Spielern

Wenn ein Spieler eine "throw" Powerup hat und versucht eine Bombe zu setzen während er sich in einer Bombe befindet wird die throw function der Bombe mit dem Feld 2 Felder Weiter vom Spieler in Blickrichtung aufgerufen

```
elif stats.can_throw and in_bomb:
    var tilemap = get_parent().get_parent().get_node("TileMap")
    var coords = tilemap.world_to_map(self.position - tilemap.position)
    in_bomb.throw(coords+self.viewing_direction*2)
```

Wenn ein Spieler ein "kick" Powerup hat und mit einer Bombe kollidiert, wird die move function der Bombe aufgerufen mit der Blickrichtung des Spielers.

```
func _on_PlayerIntersection_body_entered(body):
    if body.is_in_group("Players") and body.stats.can_kick and not $CollisionShape2D.disabled and get_collision_mask_bit(0):
        if body.stats.has_mirror:
            move(-body.viewing_direction)
        else:
            move(body.viewing_direction)
```