CMPE 114 PROJECT

2D Top Down RPG

Design Report

A picture containing grass, fabric

Description automatically generated

|  |  |
| --- | --- |
| Yusuf Tan | A picture containing insect, linedrawing  Description automatically generated |
| Umuthan Özel | Letter  Description automatically generated |
| Alican Cennetkuşu |  |
| Zain Qasim | A picture containing insect  Description automatically generated |

Table of Contents

[General principles for our game engine 3](#_Toc102183393)

[Chart of Class Hierarchy 4](#_Toc102183394)

[Descriptions of all methods and variables on each class 5](#_Toc102183395)

[Work Division 10](#_Toc102183396)

[1. Umuthan Özel 10](#_Toc102183397)

[2. Zain Qasim 10](#_Toc102183398)

[3. Yusuf Tan 10](#_Toc102183399)

[4. Alihan Cennetkuşu 10](#_Toc102183400)

# General principles for our game engine

Our game engine works by dividing every code to 4 types. These types are all stored in their respective packages. To start;

1. **Component:** Components are what makes any object do something. This includes things like adding physics interactions to our entities, map creating and rendering for our window and such.
2. **Entity:** Entities include everything that can move, be interacted with, and can be expanded with the use of components. Entities include classes like Player, Enemy classes, and such. They do not do much on their own, but for example, just adding a collider to an Enemy class, suddenly that enemy becomes interactable.
3. **Resource:** Resources are generally custom created data structures that don’t do anything on their own. They are just a blueprint for other classes to store complex data like animations, tiles, collision results, etc.
4. **Util:** Utils are things that help with other classes but are not used on their own. For example, to store position of an object, instead of having two floats, we have Vector2 util class. This class stores two floats for position, but also has some methods like normalization in them to help with anything.

These classes are all implemented 100% by ourselves. Only library we use is Swing for Java to create Windows frames. Other than that, from game loop to rendering, physics to input handling, everything is done all by us.

# Chart of Class Hierarchy

|  |  |
| --- | --- |
| **src.component.AnimationManager** | |
| Methods | |
| **void** CreateAnimation(String \_name, int \_startFrame, int \_speed) | |
| Creates Animation class and adds it to its list. | |
| **void** SwitchAnimation(int \_index) | |
| Changes animation. | |
| **void** update() | |
| Gets called from the GamePanel, updates current frame of animation. | |
| Variables | |
| **ArrayList<Animation>** animations | Stores the animation frames. |
| **BufferedImage** currentImage | Stores the current frame of animation to be rendered on the screen. |
| **int** currentAnimation | Index of current animation. |
| **Int** currentFrame | Index of current frame of animation. |
| **int** time | Keeps track of time. |

# Descriptions of all methods and variables on each class

|  |  |
| --- | --- |
| **src.component.Camera** | |
| Methods | |
| Camera(int xPosition, int yPosition) | |
| Constructor without a target. | |
| Camera(int xPosition, int yPosition, Entity target) | |
| Constructor with a target attached. | |
| **void** update() | |
| Gets called from the GamePanel, updates position of the camera. | |
| **Boolean** IsInsideBoundaries(Vector2 \_pos, float \_offset) | |
| Checks if given position is inside the view boundaries. | |
| **void** SetTarget(Entity \_target) | |
| Sets the target as the given entity. | |
| Variables | |
| **Vector2** position | Stores the position |
| **Entity** target | Target entity to follow. |

## 

|  |  |
| --- | --- |
| **src.component.Collider** | |
| Methods | |
| **Boolean** IsColliding() | |
| Checks if collision is happening | |
| **ArrayList<Collision>** CheckForCollisions() | |
| Checks for collisions and returns them with a list of Collision instances. | |
| Variables | |
| **Entity** parent | Entity that collider is attached to. |
| **int** xOffset, yOffset, sizeX, sizeY | Stores the position and size of the collider. |
| **Vector2[]** posCheck | Positions to check collision for. |
| **static ArrayList<Collider>** colliderList | All colliders on the scene. |
| **ArrayList<Collision>** collisions | All the collision data at that spesific frame. |

|  |  |
| --- | --- |
| **src.component.KeyHandler** | |
| Methods | |
| **void** KeyPressed(KeyEvent e) | |
| Set the state of a key press with keyboard input. | |
| **void** KeyRelased(KeyEvent e) | |
| Set the state of a key press with keyboard input only when key is relased. | |
| Variables | |
| **List<KeyBehaviour> keyList** | List of key states. |

|  |  |  |
| --- | --- | --- |
| **src.component.TileManager** | | |
| Methods | | |
| **void** LoadTiles() | | |
| Loads tiles from sprite folder. | | |
| **void** AddTile(String \_name, int \_index, boolean \_isSolid) | | |
| Creates a tile. | | |
| **void** LoadMap() | | |
| Loads map from map txt file. | | |
| **Tile** WorldCoordinateToTile(Vector2 pos) | | |
| Converts world units to a tile. | | |
| **Vector2** WorldToTileCoordinate(Vector2 pos) | | |
| Converts world units to tile units. | | |
| Variables | | |
| **int** height, width | Stores the dimensions of the map. | |
| **Tile[][]** map | Map data stored as a 2D array of tiles. | |
| **ArrayList<Tile>** tileList | All the blueprints for the tiles. | |
| **src.entity.Enemy** | |
| Methods | |
| **void** start() | |
| Start operations. | |
| Variables | |
| Nothing implemented yet. |  |

|  |  |
| --- | --- |
| **src.entity.Entity** | |
| Methods | |
| **void** start() | |
| Start operations. | |
| **void** update() | |
| Updates the entity properties. | |
| **void** render() | |
| Renders the player. Uses AnimationManager component to do this. | |
| Variables | |
| **String** name | Name of the entity. |
| **Vector2** position | Position of the entity. |
| **float** speed | Speed of the entity. |
| **enum Direction** entityDirection | Direction of the entity. |

|  |  |
| --- | --- |
| **src.entity.Player** | |
| Methods | |
| **void** start() | |
| Start operations. | |
| **void** update() | |
| Updates the entity properties. | |
| **void** render() | |
| Renders the player. Uses AnimationManager component to do this. | |
| **void** InputHandler() | |
| Handles the keyboard input. | |
| **void** UseWeapon() | |
| Uses the current weapon - Not implemented yet | |
| Variables | |
| **Boolean** isMoving | Name of the entity. |

|  |  |
| --- | --- |
| **src.resource.Animation** | |
| Variables | |
| **int** startFrame | Start frame index. |
| **int** endFrame | End frame index. |
| **int** speed | Speed of the animation. |
| **int** frameCount | Total frame count. |

|  |  |
| --- | --- |
| **src.resource.Animation** | |
| Variables | |
| **int** startFrame | Start frame index. |
| **int** endFrame | End frame index. |
| **int** speed | Speed of the animation. |
| **int** frameCount | Total frame count. |

|  |  |
| --- | --- |
| **src.resource.Collision** | |
| Variables | |
| **Vector2** position | Position of collision. |
| **Entity.Direction** direction | Direction of collision. |
| **enum CollisionType** collisionType | If collision is entity or tile. |
| **Entity** entity | Collided entity. |
| **Tile** tile | Collided tile. |

|  |  |
| --- | --- |
| **src.resource.Tile** | |
| Variables | |
| **BufferedImage** image | Image of the tile. |
| **Boolean** solid | If tile is solid or passable. |
| **String** name | Tiles name. |

|  |  |
| --- | --- |
| **src.util.Vector2** | |
| Methods | |
| **Vector2** Normalize() | |
| Normalizes vector by dividng it to its length. | |
| **void** Add(Vector2 \_vectorToAdd) | |
| Adds two vectors. | |
| Variables | |
| **float** x, y | Position values. |

|  |
| --- |
| **src.util.RenderUtils** |
| Methods |
| **static void** DrawSprite(Vector2 pos, BufferedImage image, Boolean checkIfOnBoundary, Graphics2D g2D) |
| Draw sprite on screen. |
| **static void** DrawRect(Vector2 pos, int sizeX, int sizeY, Color color, Graphics2D g2D) |
| Draw rectangle on screen. |
| **static void** DrawLine(Vector2 pos1, Vector2 pos2, Color color, Graphics2D g2D) |
| Draw line between two given positions. |

# Work Division

### Umuthan Özel

Core Game Engine development. Currently working on the component system and building the base that the game will eventually build upon.

### Zain Qasim

Character creation screen, main menu interface, and everything related to interfaces in the game.

### Yusuf Tan

Enemy AI, enemy design, combat patterns and weapon functionalities.

### Alican Cennetkuşu

Map generation, map and level design, all things related to the adventuring part of the game.