L2 Documentation

L.O.S.T

Logically Oriented Software Technology

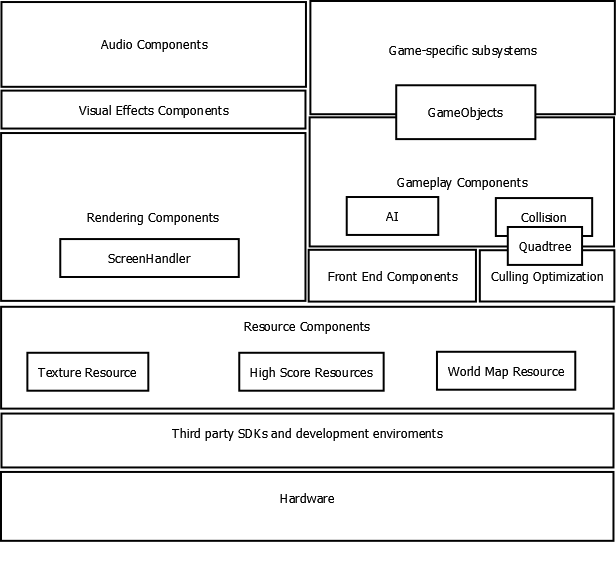
Rasmus Tilljander - rati10@student.bth.se

Erik Hörlin - erhd10@student.bth.se

Calle Ketola - cake10@student.bth.se

Kim Hansson - kiha10@student.bth.se

# Updated Architecture from L1



We added some minor components to the Architecture, things that we already talked about during the L1 but are now properly defined and ready to be implemented.

- The rendering now holds a ScreenHandler to switch between screens.

- The gameplay now includes an AI as well as Collision. Collision uses a Quadtree which is also used for Culling Optimization.

- Gameplay as well as Game-specific subsystems both share our GameObjects which includes Candy, S-Candy, ghosts etc.

# J:\FOOT_Pacman\Reports\L2\BilderUML\L.O.S.T Class diagram.pngClass Diagram

# Class Descriptions

## Main

The main class of the game. It will create the Window's handle as well as maintain the message loop which runs the update and draw for the renderer for the project.

## Renderer

The rendering component for the game. Will handle the creating of the d3dDevice as well as other DirectX components. It then renders and updates the ScreenHandler. This part will be reused from an older project.

## AudioHandler

Class for handling Fmod to play different sounds and music-loops. This will be a Singleton class to be shared over all screens. This is a third-party component.

## Camera

Represents the view of the player. It will be a first person camera so that the player can feel as if he is Pacman himself. This part will be reused from a older project.

## KeyboardInputHandler

Will handle the key input from the keyboard so that the different screens can do checks for whether a key has been pressed or not.

### Key(Part of KeyboardInputHandler)

An enumerator used to represent different keys that are used by the game.

## ScreenHandler

Draws the current screen to be viewed by the player. Holds the main part of the update loop which is called from the message loop. It checks continually to see if the current screen needs to be swapped for another one. The whole screen system will be built using previous knowledge and experience of a similar system but no code will be reused. The ScreenHandler will implement an observer pattern.

### GameScreenState(Part of ScreenHandler)

An enumeration used to swap between the different screens in the ScreenHandler. Different events in the screens will trigger them to make changes to the update loop they inherit from the BaseScreen. This will then be checked by the ScreenHandler and the current screen will be updated.

## BaseScreen

Abstract class from which the other screens inherit basic functions(the GameScreen and DeathScreen inherit from BaseGameScreen which in turn inherits from the BaseScreen).

## BaseGameScreen

Abstract class from which the GameScreen and the DeathScreen will inherit certain functions.

## GameScreen

Handles the game play view when the player is playing a level. Holds the WorldHandler and the CollisionHandler.

## MenuScreen

Handles the view of the menu that the player navigates before starting a new game. Holds MenuObjects used for interacting with the menu.

## DeathScreen

When Pacman gets eaten by a ghost this screen will be displayed to show the 2D-world and Pacman haunting the ghosts.

## EndScreen

Once the game ends this screen will be displayed to let the player enter his score into a high score list.

## MapScreen

When the game is paused this screen will be displayed, showing an overview map of the game level.

## GameTimer

Timer used for calculating the elapsed game time in seconds and milliseconds. This part will be reused from an older project

## HUD

The front-end component for the game. This will be used to display game related information to the player such as, the current score, how many lives are left etc. This part will be reused from an older project.

## CollisionHandler

Class for handling all the in game collisions between any kind of GameObject.

## WorldHandler

Holds the structure of the current level, creating WallObjects to fit with the level design. Also holds the QuadTree used for partitioning the level.

## QuadTree

Quad tree structure used to divide the world for rendering optimization and collision checks. Holds Node objects.

## ShaderObject

Creates and handles the shader files and the input layout associated with each GameObject. Also used to let the Object draw itself. This part will be reused from an older project.

## GameObject

Abstract class from which the other Object-classes inherit basic functions(the player, ghost objects inherit from the NonStaticObject class which in turn inherits from GameObject).

## WallObject

Object used for representing pieces of walls.

## NonStaticObject

Abstract class from which Objects such as Ghosts and Player will inherit.

## Player

Object to represent Pacman. When the project receives input from the keyboard it will update the position in this class. The camera gets its position from this class as well.

## Ghost

Object for handling the enemy AI. Moves using a grid structure with waypoints.

## Candy

The basic candy which exists in the game. Gives the player a small number of points to his score when eaten. The level is cleared once all the candy is gone.

## SuperCandy

Special candy which changes Pacman's state so that he can eat the ghosts for extra points.

## MenuObject

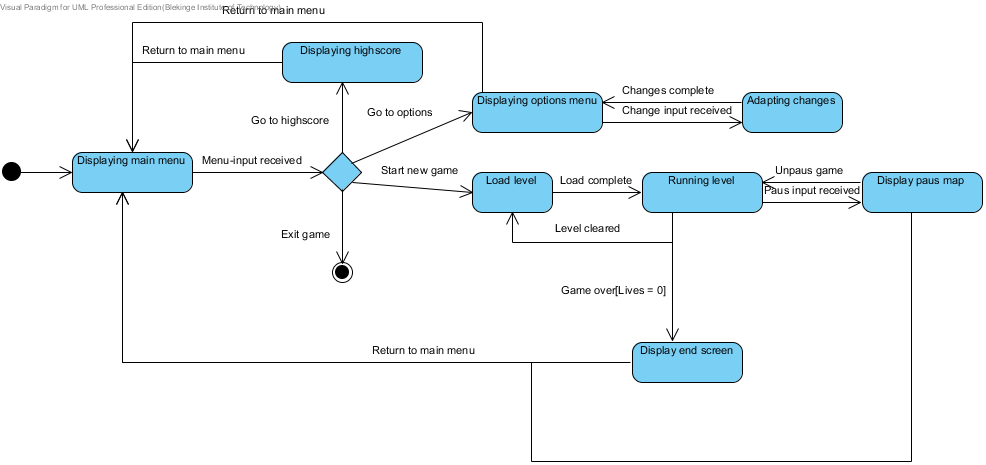
Represents menu components such as buttons used for interacting with the menu.

## TextureHandler

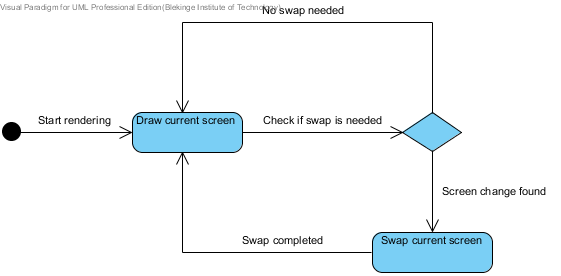
Loads the textures for objects. By having a separate handler for the textures, multiple objects that share the same texture do not have to reload it. This class will implement a singleton pattern.

# State Machine Diagrams

## Overview



## ScreenHandler



## Pacman

