N.Y TAJMS STUDIOS PROUDLY PRESENTS!

A TAJMS OPENGL PRODUCTION

PACMAN 3D

Group Members:

Eric Ahlström

John Bergsten

Lucas Holmqvist

Christian Markowicz

Konrad Öhman

Contents

Introduction

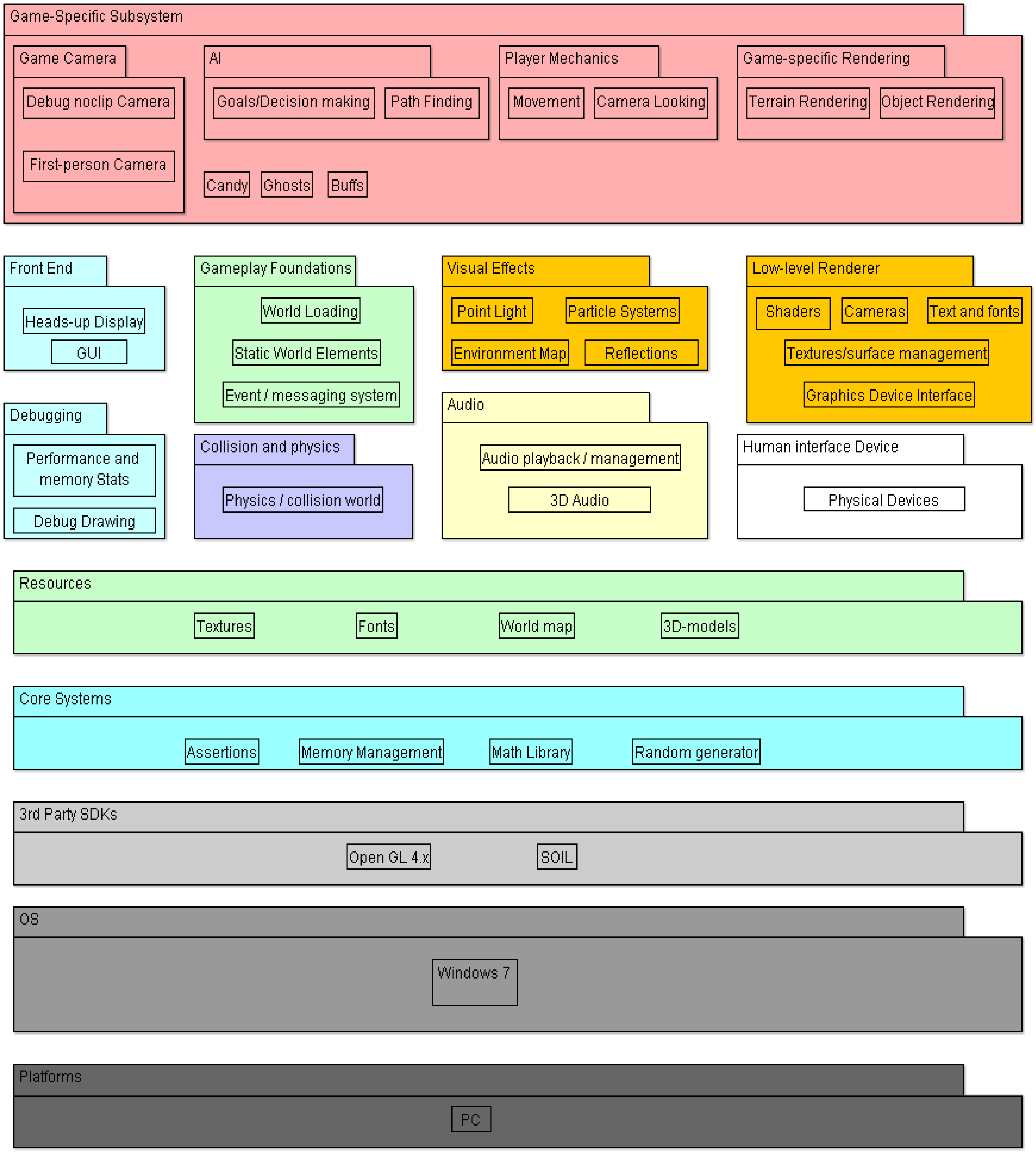
Pacman is one of those ironical video games produced during the industry’s infancy. No one calls him/herself a gamer without having at least heard of the legend that is Pacman. We are thrilled to now be tasked with the creation of our own version. This document is the basic conceptual outlines of what we hope our game will be.

Gameplay introduction

The player plays as Pacman from a first-person perspective, albeit in a 3D environment. The goal, as with the classic Pacman-games, is to eat all the candy while avoid being eaten by the ghosts. One fundamental change is the layout of the map. The player has to navigate on paths surrounded by black pits which Pacman can fall down into and die.

Aside from the dangers of falling down from the paths, Pacman will starve if he fails to continually eat candy. This has consequences: Pacman’s movement speed will be reduced and the light will begin to fade, which will make spotting and running away from ghosts significantly harder.

Architecture Diagram



Game-specific Subsystem

This block contains components that are specific for this game, such as the candy and ghosts.

Camera contains two cameras: a noclip-camera to easily fly around the world for debug purposes and a first-person game-camera for the final gameplay.
AI directs the ghosts' movement by sending a path to each ghost. It also tells the ghost whether to flee from Pacman or to chase him.
Player mechanics manages play-inputs from the keyboard and mouse which are then used to control Pacman and for navigation of menus.
Game-specific Rendering renders the terrain and different objects.
Candy are small pieces of candy which Pacman eats.
Ghosts are the main enemy. They follow a path given by the AI, based on their state and Pacman's position.
Buffs changes the state of Pacman. The most basic buff is that he can eat the ghosts for a brief period of time.

Front-end System

Contains HUD and GUI. These are things that don't interact with the gameplay but rather shows info that is useful to the player and directs the player to different places of the program.

HUD (Heads Up Display) contains essential information such as lives and score which is then presented on the screen during gameplay.
GUI (Graphical User Interface) includes different menus which navigates the player to different places in the program.

Gameplay Foundations

This block contains map loading, different static objects (like spawn points) and an event/messaging system.

Map Loading reads the map and saves both a logical representation of the map for collision detection and path finding, and a graphical representation that is sent to the graphic interface.
Static World Elements are elements/objects that are in the same place throughout the game, like spawn points.
Event/Messaging System gives the player information about gameplay states, for example when the ghosts changes state from fleeing to chasing.

Visual Effects

Different lighting and special effects.

Point Light: a light source emitting light from one point. It is the main source of light in our game.
Particle Effects is a way to demonstrate small particles with different movement patterns. Explosions are represented by particle effects.
Reflections put a texture on a surface that represents the view of the surface.
Environment Map controls background textures.

Low-Level Renderer

Contains lower levels of rendering, such as shaders.

Shaders are programs that reside in the GPU. They tell the GPU how to color each pixel seen by the camera.
Cameras are matrices that convert the world into different views.
Text And Fonts are outputs to the GUI and can also be used for debugging. The fonts tell the text how to display itself.
Textures/Surface Management puts the correct texture on the right object.
Graphics Device Interface is the interface towards the low-level renderer.

Debugging

Is used to check the program code for bugs, glitches and performance issues.

Performance and memory stats keep stats about performance, such as frames per second.
Debug Drawing prints the map and objects in different ways. Can be used for drawing only the outlines of polygons.

Collision and physics

Used to check different forms of collisions.

Physics / Collision World use positions to check if collisions occur.

Audio

Plays sounds and music depending on the position of Pacman.

Audio Playback / Management controls what sounds to play and what volume it should be played at. The volume depends on the distance from the audio source and Pacman.
3D Audio makes the sound appear different in each output device.

Human Interface Device

Mouse and keyboard are the only interface devices that the player can use.

Physical Devices. The keyboard is used to control Pacman left, right, forward and backward. The mouse is used to control the camera and Pacman's direction.

Resources

Useful files for our program.

Textures: files with colors that we can attach to an object.
Fonts: information that describes the way a text should be written.
World Map stores information about how the vertices are to be constructed and how the logical map will be contracted. It is read from a .raw map.
3D-models are stored in .obj files with information on how to construct the vertices for a 3D-object.

Core Systems

These are the different libraries which help us by providing predefined methods.

Assertions check the code for logical errors like misspelling or forgotten characters.
Memory Management helps with deleting memory places that won't be used again. This makes memory loss less frequent.
Math Library provides structures and mathematical calculations.
Random Generator uses the CPU clock to give the program a seemingly random number.

3rd Party SDKs

Software programmed by a 3rd party which we make use of in our program.

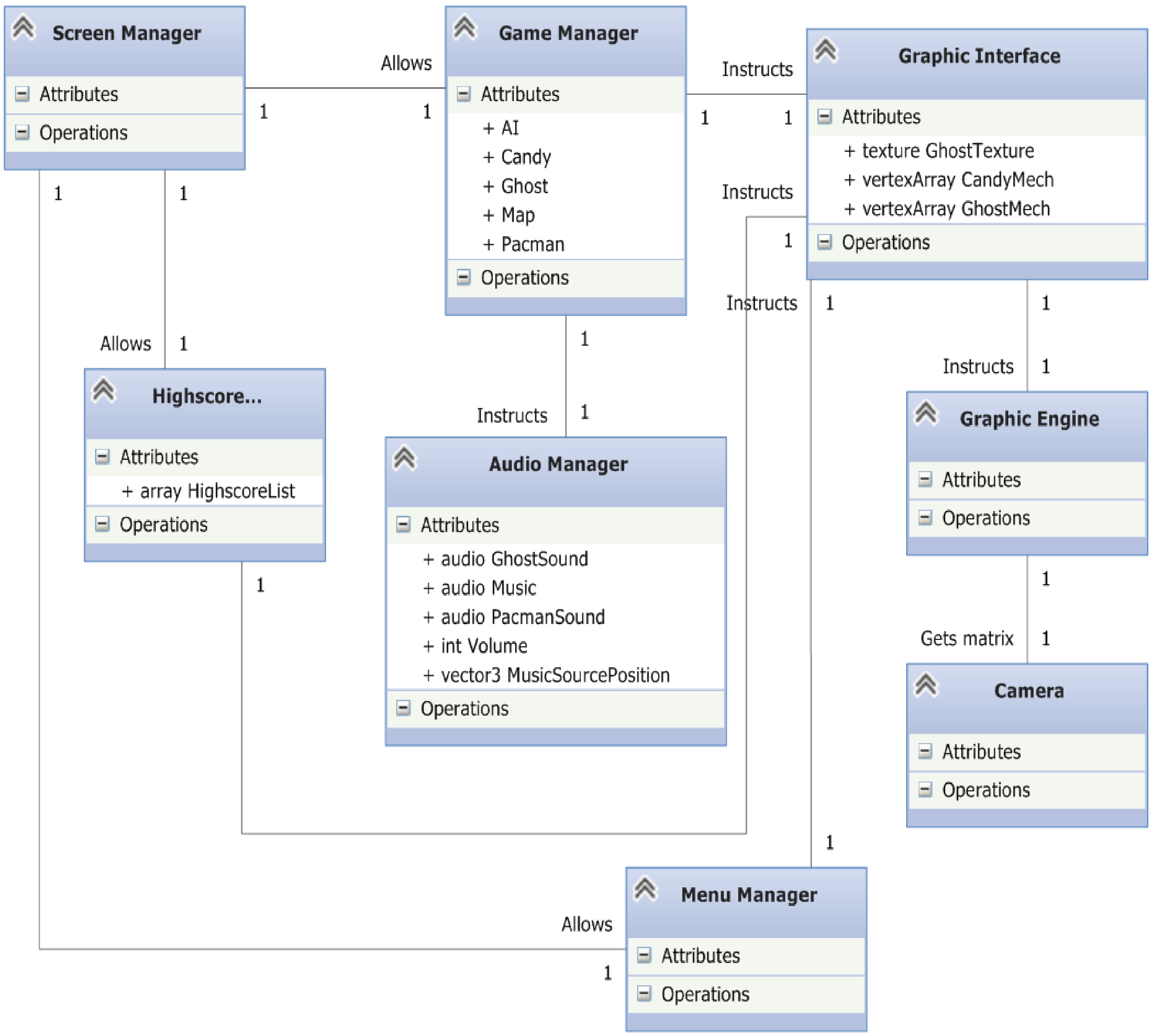
Open GL 4.x is the SDK that will be used to manage the GPU.

SOIL reads textures.

OS and Platform

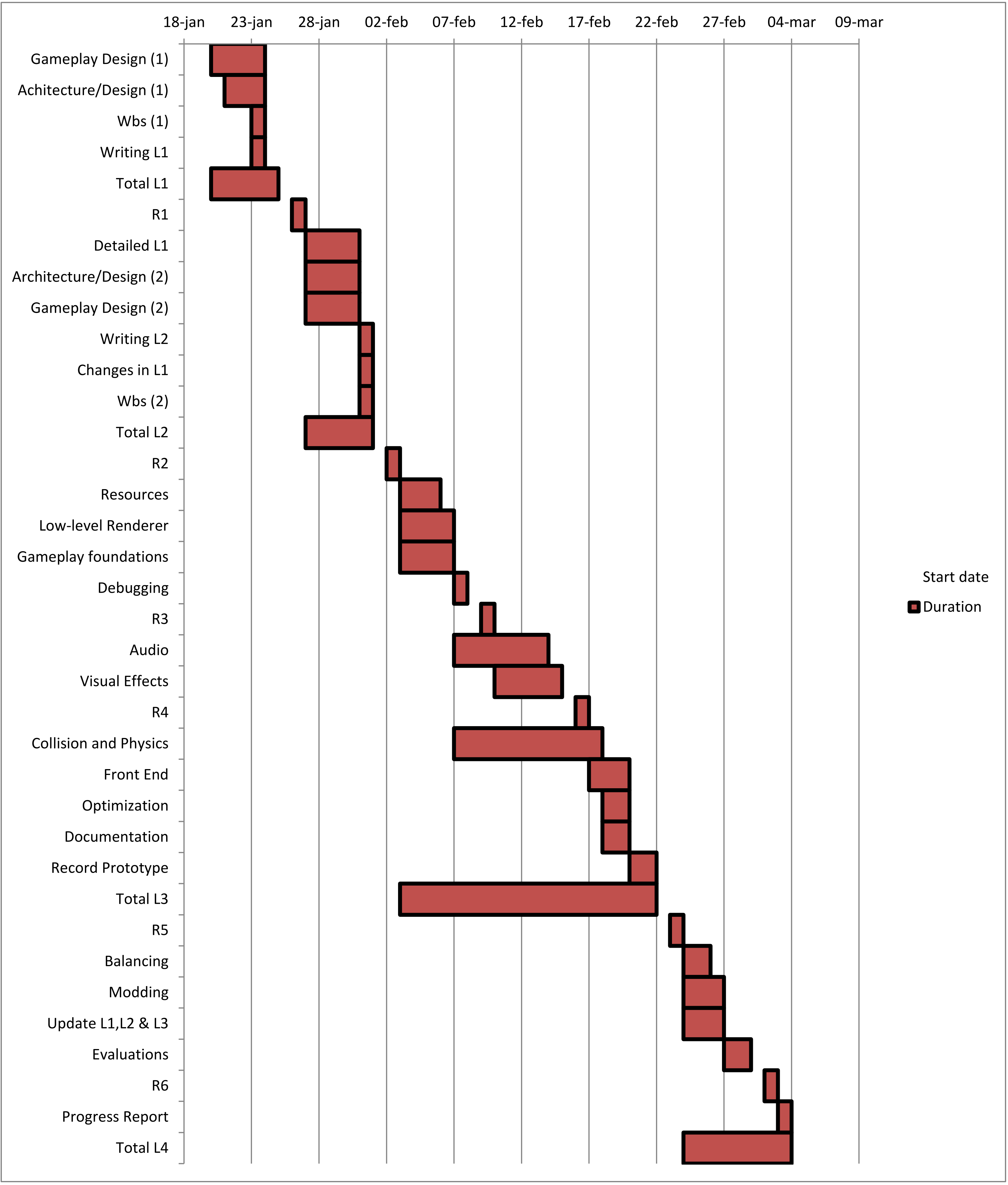
This game will be programmed for Windows 7 using a pc.

Domain Modell



Work Breakdown Structure

Task	Start Date	Duration(days)	End Date	Est. Time(h)	Act. Time(h)	Act. Time/ Est. Time(%)
Gameplay Design (1)	20-jan	4	23-jan	2	2	100,00%
Achitecture/Design (1)	21-jan	3	23-jan	58	58	100,00%
Wbs (1)	23-jan	1	23-jan	3	7	233,33%
Writing L1	23-jan	1	23-jan	12	11	91,67%
Total L1	**20-jan**	5	**24-jan**	**75**	**78**	**104,00%**
R1	26-jan	1	26-jan	10		0,00%
Detailed L1	27-jan	4	30-jan	50		0,00%
Architecture/Design (2)	27-jan	4	30-jan	5		0,00%
Gameplay Design (2)	27-jan	4	30-jan	2		0,00%
Writing L2	31-jan	1	31-jan	10		0,00%
Changes in L1	31-jan	1	31-jan	5		0,00%
Wbs (2)	31-jan	1	31-jan	2		0,00%
Total L2	27-jan	5	31-jan	**84**	**0**	**0,00%**
R2	02-feb	1	02-feb	10		0,00%
Resources	03-feb	3	05-feb	15		0,00%
Low-level Renderer	03-feb	4	06-feb	40		0,00%
Gameplay foundations	03-feb	4	06-feb	40		0,00%
Debugging	07-feb	1	07-feb	10		0,00%
R3	09-feb	1	09-feb	10		0,00%
Audio	07-feb	7	13-feb	50		0,00%
Visual Effects	10-feb	5	14-feb	50		0,00%
R4	16-feb	1	16-feb	10		0,00%
Collision and Physics	07-feb	11	17-feb	35		0,00%
Front End	17-feb	3	19-feb	30		0,00%
Optimization	18-feb	2	19-feb	20		0,00%
Documentation	18-feb	2	19-feb	20		0,00%
Record Prototype	20-feb	2	21-feb	20		0,00%
Total L3	03-feb	19	21-feb	**360**	**0**	**0,00%**
R5	23-feb	1	23-feb	10		0,00%
Balancing	24-feb	2	25-feb	20		0,00%
Modding	24-feb	3	26-feb	30		0,00%
Update L1,L2 & L3	24-feb	3	26-feb	15		0,00%
Evaluations	27-feb	2	28-feb	50		0,00%
R6	02-mar	1	02-mar	10		0,00%
Progress Report	03-mar	1	03-mar	25		0,00%
Total L4	24-feb	8	03-mar	**160**	**0**	**0,00%**


18-jan
23-jan
28-jan
02-feb
07-feb
12-feb
17-feb
22-feb
27-feb
04-mar
09-mar
Gameplay Design (1)
Achitecture/Design (1)
Wbs (1)
Writing L1
Total L1
R1
Detailed L1
Architecture/Design (2)
Gameplay Design (2)
Writing L2
Changes in L1
Wbs (2)
Total L2
R2
Resources
Low-level Renderer
Gameplay foundations
Debugging
R3
Audio
Visual Effects
R4
Collision and Physics
Front End
Optimization
Documentation
Record Prototype
Total L3
R5
Balancing
Modding
Update L1,L2 & L3
Evaluations
R6
Progress Report
Total L4
Start date
Duration

Class Diagram

The program is separated into two class diagrams, one for the game logic and one for the graphic.

Game logic class diagram

Graphic class diagram

References

We have used the course main literature as a template for our architecture diagram. We have removed some of the content that they had since we found it unnecessary for this project. [1] For example we decided to not include skeletal animations on the basis that it is pointless in such a simple game as Pacman.

[1] Game Engine Architecture, Jason Gregory, 2009 Taylor and Francis group, p.29