N.Y TAJMS STUDIOS PROUDLY PRESENTS!

A TAJMS OPENGL PRODUCTION

IN ASSOCIATION WITH BTH

PACMAN 3D

Group Members:

Eric Ahlström

John Bergsten

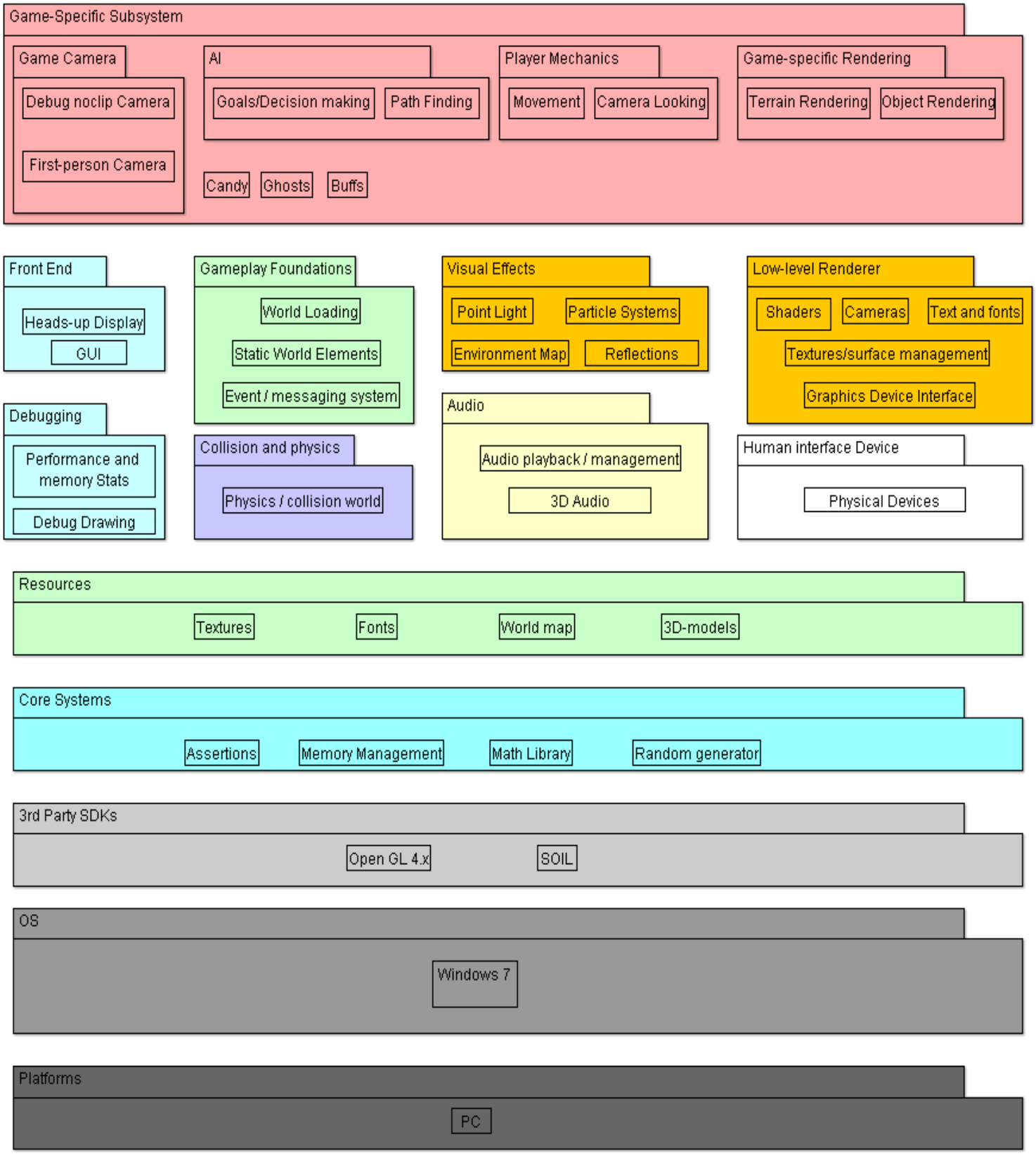
Lucas Holmqvist

Christian Markowicz

Konrad Öhman

Contents

Architecture Diagram



Game-specific Subsystem

This block contains components that are specific for this game, for example candy, ghost, camera etc.

Camera contains functions to build a first person camera and a free moving camera. The free moving camera is for debugging while the first person camera is for the final product.
AI directs the ghosts movement by sending a map of directions to each ghost. It also tells the ghost whether to flee from Pacman or to chase him.
Player mechanics uses the player input, from the mouse and keyboard, to control the direction of the camera and Pacmans position.
Game-specific Rendering holds info on how to render the terrain and the different objects.
Candy holds a position to where the candy is.
Ghosts are the main enemy. They hold a map on where to go which they get from the AI and a state, whether to chase or flee.
Buffs change the state of Pacman. The most basic of buffs is that he can eat the ghosts for a specific amount 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 is short for Heads Up Display. It contains different information like score, HP, if Pacman can eat the ghosts etc.
GUI is short for Graphical User Interface. This 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 a event/messaging system.

Map Loading reads the map and saves both a logical representation of the map, for collision detection and pathfinding, and a graphical representation that is sent to the graphic interface for on-screen representation.
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

The visual effects of the program includes point light(s), different particle effects, reflections and an environment map.

Point Light is a source of light with a direction and a source 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 information that draws the graphic.

Shaders are programs that reside in the gpu. They tell the gpu how to paint every pixel on the screen.
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. This includes surfaces.
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. For example fps, memory loss etc.
> **Debug Drawing** prints the map and objects in different ways. Can be used for drawing the outlines of triangles only.

Collision and physics

Functions that are used for gameplay events and changes for example Pacman collides with an enemy and is killed.

> **Physics / Collision World** use positions to check if a move is prohibited.

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, up and down. 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. Uses a .raw file to store the information.
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 that helps us with predefined methods.

Assertions check the code for logical error 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 gives structures and mathematical calculations on more advanced variables.
Random Generator uses the CPU clock to give the user a seemingly random number.

3rd Party SDKs

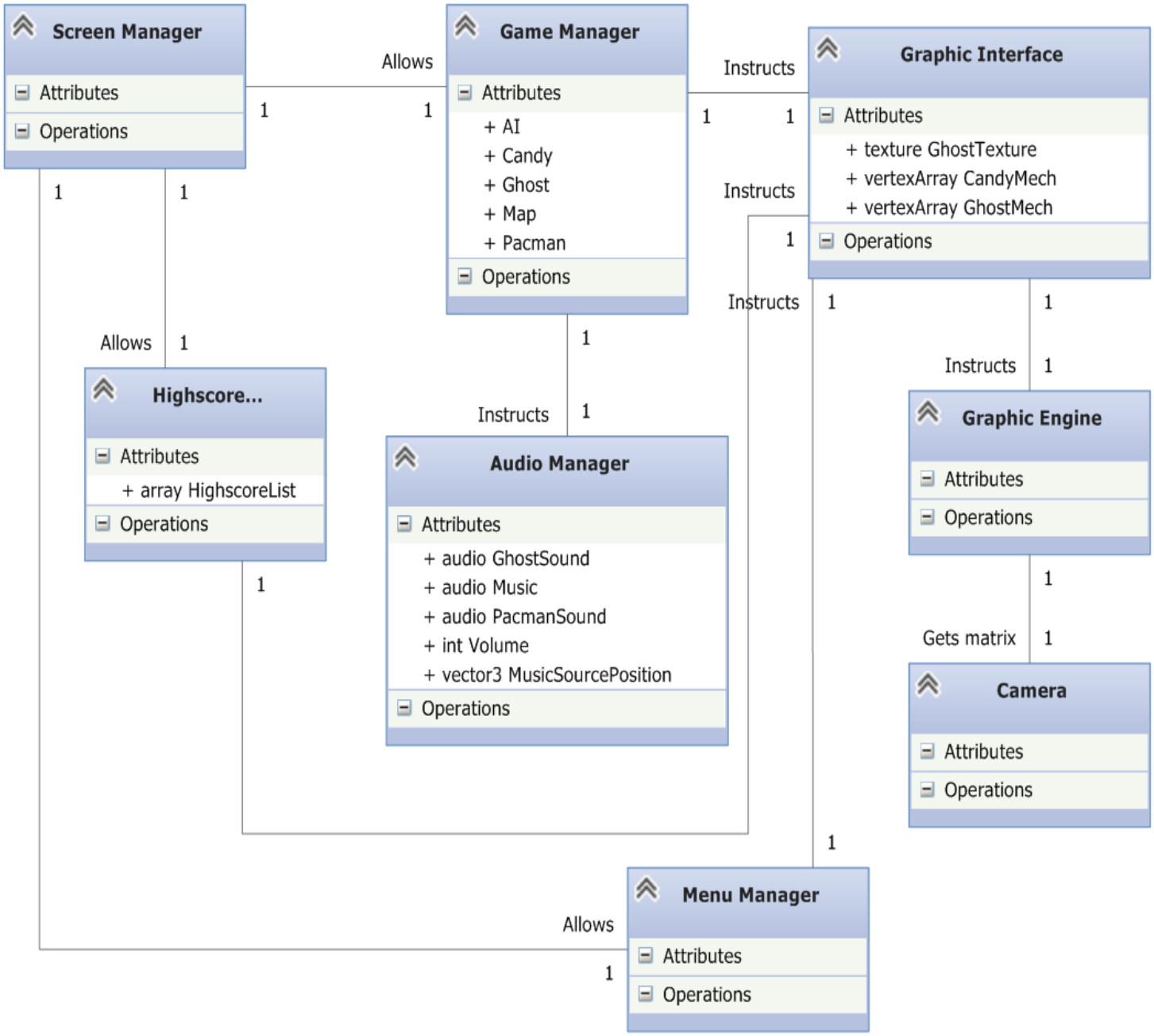
These are the libraries, dlls and .h files needed to do more advanced stuff. These are programmed by a 3rd party.

Open GL 4.x is the SDK that will be used to manage the GPU. It, among other stuff, sends buffers to the GPU and tells it what shaders are to be used.
SOIL helps with reading in 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	4	133,33%
Writing L1	23-jan	1	23-jan	12	10	83,33%
Total L1	**20-jan**	5	**24-jan**	**75**	**74**	**98,67%**
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%**