GameLoft Xtream programming challenge 2010 round 1

Candidate: Le Hoang Quyen

TECHNICAL DESIGN

OUTLINE

1.Introdution:

-When GameLoft had announced the 2nd Xcode challenge that will mostly focus on 3D & openGL,I had an idea about writing a small game engine that can be reused in my future projects.So I decided to start writing a part of my engine alongside with the 1st round required program.

-When I finished the 1st round required program,my engine was nowhere near it should.But it had enough basic features for a mini game such as this program.

-My visual studio solution for 1st round program contains 5 main projects (4 first projects is parts of my engine):

+ Project **Renderer** - Creates a dynamic link library that implements a renderer API that hides the underlying graphics API (such as openGL or Direct3D).API independence can be achieved through this API definition . For the scope of this challenge , openGL will be the graphics API used by this Renderer implementation in this project.

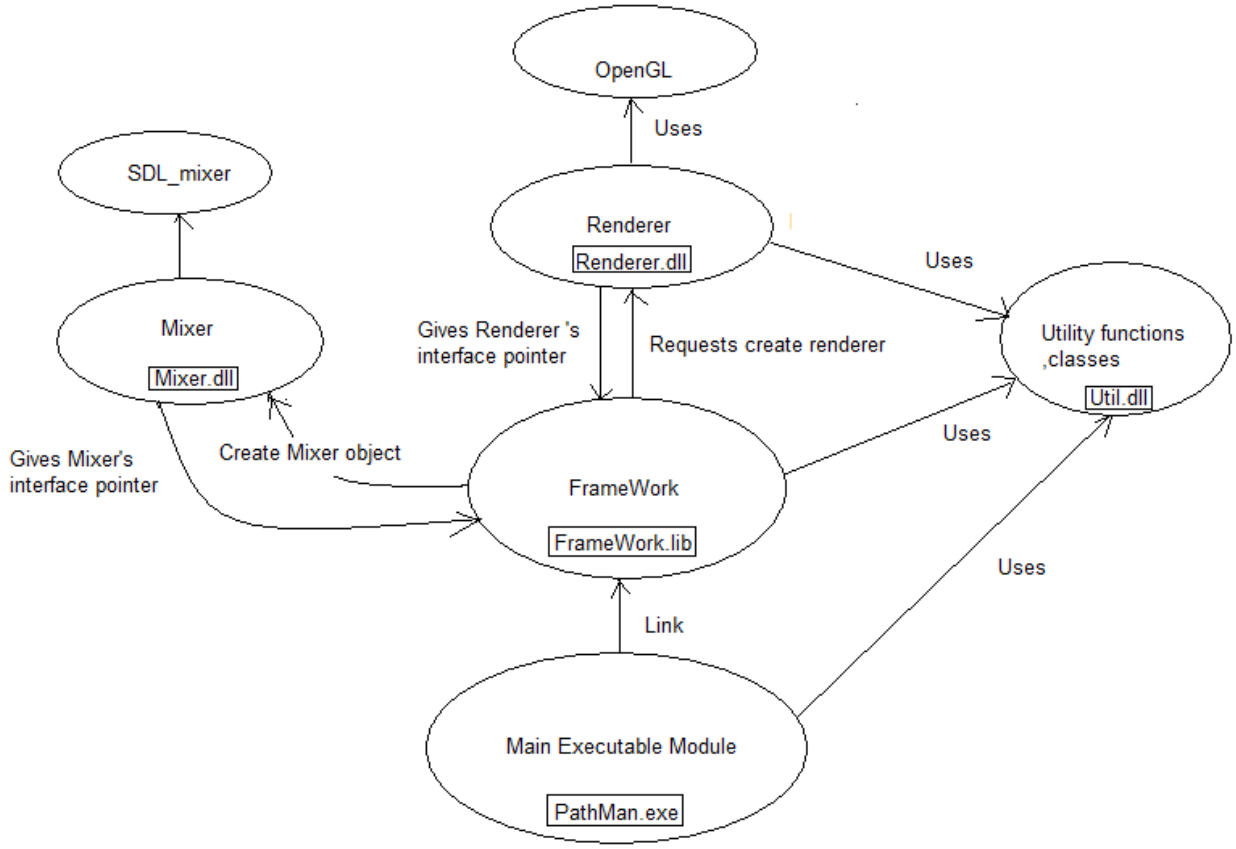
+ Project **Util** – Creates a dynamic link library that implements some utility functions,class template such as 3D Vector,matrix math,package manager ,shared pointer.v.v..v.v

+ Project **Mixer** – Creates a dynamic link library that implements a sound mixer API.The Sound Mixer implementation in this project uses SDL_mixer.

+ Project **FrameWork** – Creates static library that implements a FrameWork class which manages some common tasks for a simple game,such as keyboard mouse input,.v.v. graphics,sound,GUI management..v.v.v This class's instance can only be created as a singleton when application starts,and will be destroyed when application exits.

+ Project **PathMan** – Creates main executable module of the program. Game flow,logic,AI ..v.v. are implemented in this project.

2.Diagram:



3.Project "Renderer":

As introduced in 1st section, this project creates a dynamic link library that implements a renderer API that hides the underlying graphics API (such as openGL

or Direct3D).API independence can be achieved through this API definition . For the scope of this challenge , openGL will be the graphics API used by this Renderer implementation in this project.

-Some main classes implemented in this project:

+RendererImp – implements interface Renderer.This class is basically a wrapper around openGL functions.It can reduce some duplicated openGL's state changes ,supports some 2D rendering features alongside the core 3D rendering features.

+TextureManager – supports texture loading from file or memory,resizing image if its width & height are not power of two and end-user PC not supports non power of two texture.Can reduce some duplicated texture binding too.

+Font – supports creating and rendering text with a bitmap font.Info of each glyph in font is read from text file.Each info includes name of the glyph,rectangle area of the glyph icon in image.

+StaticVertexBuffer & StaticIndexBuffer – supports creating static vertex & index buffer object if buffer object is supported in end-user PC 's openGL version or just creating vertex & index array in case buffer object is not supported.

4.Project "FrameWork":

As introduced in 1st section, this project creates static library that implements a FrameWork class which manages some common tasks for a simple game,such as keyboard mouse input,.v.v. graphics,sound,GUI management..v.v.v This class's instance can only be created as a singleton when application starts,and will be destroyed when application exits.

-Main Classes implemented in this project:

+FrameWork- manages some basic tasks about graphics,sound,input,multithreading,freeing resources upon exit,.v.v..v.v

+AnimController – template class can be used in some simple animation .It holds information of each pose in animating loop.

+VariableManager – manages a list of variables' values loaded from text file .Supported variable types are bool,integer ,unsigned integer ,float ,array of 2/3/4 integers , array of 2/3/4 floats .

+GUImanager – manages list of GUI items such as background image,buttons,texts,images.Each button can be assigned a callback function that is triggered when button is clicked.

+MeshManager – support creating mesh from OBJ file and rendering it.

5.Levels in game:

-Each level's board can be created from information read from text file (either directly in hard disk or in package file) .Information includes a array of lines ,in that,each line represents a row in board, Each character in line represents a tile in board.

if character is '0',it denotes an invisible tile.

If character is '1',it denotes a normal tile.

if character is '2',it denotes the starting tile of main char.

if character is '3',it denotes a coin is standing at this tile

if character is '4',it denotes the starting tile of a ghost

if character is '5',it denotes a box is standing at this tile

6.Collision dectection:

No special math or physic is applied here,in my game ,collision occurs only when 2 objects are in the same tile of board.

7.AI:

-In this game, there are 2 AI behaviors : wandering & chasing

When the player 's main char is not in a ghost 's range of sight,that ghost will wander randomly in board.

When the player's main char is in a ghost's range of sight ,that ghost will chase player util it catches main char or main char is out of sight.

Chasing algorithm used in this game is A* path finding.

~End~