Cloudspace



Revision: 0.0.1

Overview

Theme / Setting / Genre

2D Sidescroller/ Space Shooter

Core Gameplay Mechanics Brief

- Ship Upgrades

- random weapon upgrades

- Bullet Hell

- Avoiding Unkillable Objects

Targeted platforms

- PC

Monetization model (Brief/Document)

- Monetization Type

- Ad Driven, Micro-transactions

Project Scope

- <Game Time Scale>

-Cost = Free App Game

- Time Scale = 6 months

- <Team Size>

- <Core Team>

- Erik Enos

- Programming

- $20/h

- <Core Team>

- Fyodor

- Programming

- $20/h

- <Core Team>

- Nick F.

- Programming

- $20/h

- <Licenses / Hardware / Other Costs>

- <Total Costs with breakdown>

Influences (Brief)

- <Influence #1>

Galaga

- Cloudspace is heavily influnced by Galaga's simple yet perfect 2D sidescolling and bullet hell.

- <Influence #2>

Gradius

Our Game's art in heavily influnced by Gradius

- <Influence #3>

R-Type

- one of the main influncing is the Weapon system and the enviroment of the Levels.

The elevator Pitch

CloudSpace is the fastest, most bullet hell SpaceShooter ever made.

Project Description (Brief):

<Two Paragraphs at least>

<No more than three paragraphs>

Project Description (Detailed)

<Four Paragraphs or more If needs be>

<No more than six paragraphs>

What sets this project apart?

<Reason #1>

Fast paced space shooter

<Reason #2>

unique Music

<Reason #3>

Our new take on 2d Art design

- <Reason #4>

Massive Leveling system

Core Gameplay Mechanics (Detailed)

- <Core Gameplay Mechanic #1>

- <Details>

The Ship upgrade system will allow the player to add better hulls to increase max health, upgrade ship guns for different fire rates and damage, upgrading boosters will allow the player to incresse speed.

- <How it works>

In between level the player with be showen the ship in the current load out. From there the player will be able to select parts of the ship to upgrade or swap out from the garage

- <Core Gameplay Mechanic #2>

- <Details>

Random upgrade/power ups in the level to add fire rate up, damage up, double shot/triple shot.

- <How it works>

While the player plays the game they will destroy enemies and they with randomly drop temp power ups adding to the players damage/fire rate and speed.

- <Core Gameplay Mechanic #3>

- <Details>

Bullet Hell adds to the level of difficulty in the game.

- <How it works>

low level enemies with have very simple attack patterns. later enemies will get hardly and less peredictable making the screen fill with bullets to doged

Story and Gameplay

Story (Brief)

The main Charater in a space bounty hunter looking for his next bounty.

Story (Detailed)

Gameplay (Brief)

Shoot enemies, collect power ups and complete the level boss to collect bounty

Gameplay (Detailed)

The player will have to fight and dodge there way though waves of enemies collecting power ups and incressing there xp. At the end of each level there will be a mini boss fight followed by the LEVEL BOSS. The player will recive a big cash reward and be able to spead it on the ship.

Assets Needed

- 2D

- Textures

- Environment Textures

- Characters List

Player Character

Enemy\_1(Pawn)

Enemy\_2(Rook)

Enemy\_3(Knight)

Enemy\_4(Bishop)

MiniBoss(Queen)

LevelBoss(King)

- Environmental Art Lists

Background Stars

Planents

- Sound

- Sound List (Ambient)

- Outside

Level 1 music

Level 2 music

Level 3 music

Level 4 music

- Inside

Level 1 music(Boss theme)

Level 2 music(Boss theme)

Level 3 music(Boss theme)

Level 4 music(Boss theme)

- Sound List (Player)

- Character Movement Sound List

Forward thoust sound

Engine slowing down

- Character Hit / Collision Sound list

Regular bullet hit sound

laser bullet hit sound

rocket hit sound

player collision with enemy sound

player collision with rock sound

- Character on Injured / Death sound list

Ship beeping sound/ AI warning sound

- Animation

- Environment Animations

Stars moving across the screen

plantets moving

shooting stars

- Character Animations

- Player

Player rotating

Player shooting

- NPC

enemy rotating

enemy shooting

Schedule

- <Object #1>

- Time Scale

- Milestone 1

- Milestone 2

- Etc.

- <Object #2>

- Time Scale

- Milestone 1

- Milestone 2

- Etc.

- <Object #3>

- Time Scale

- Milestone 1

- Milestone 2

- Etc.

- <Object #4>

- Time Scale

- Milestone 1

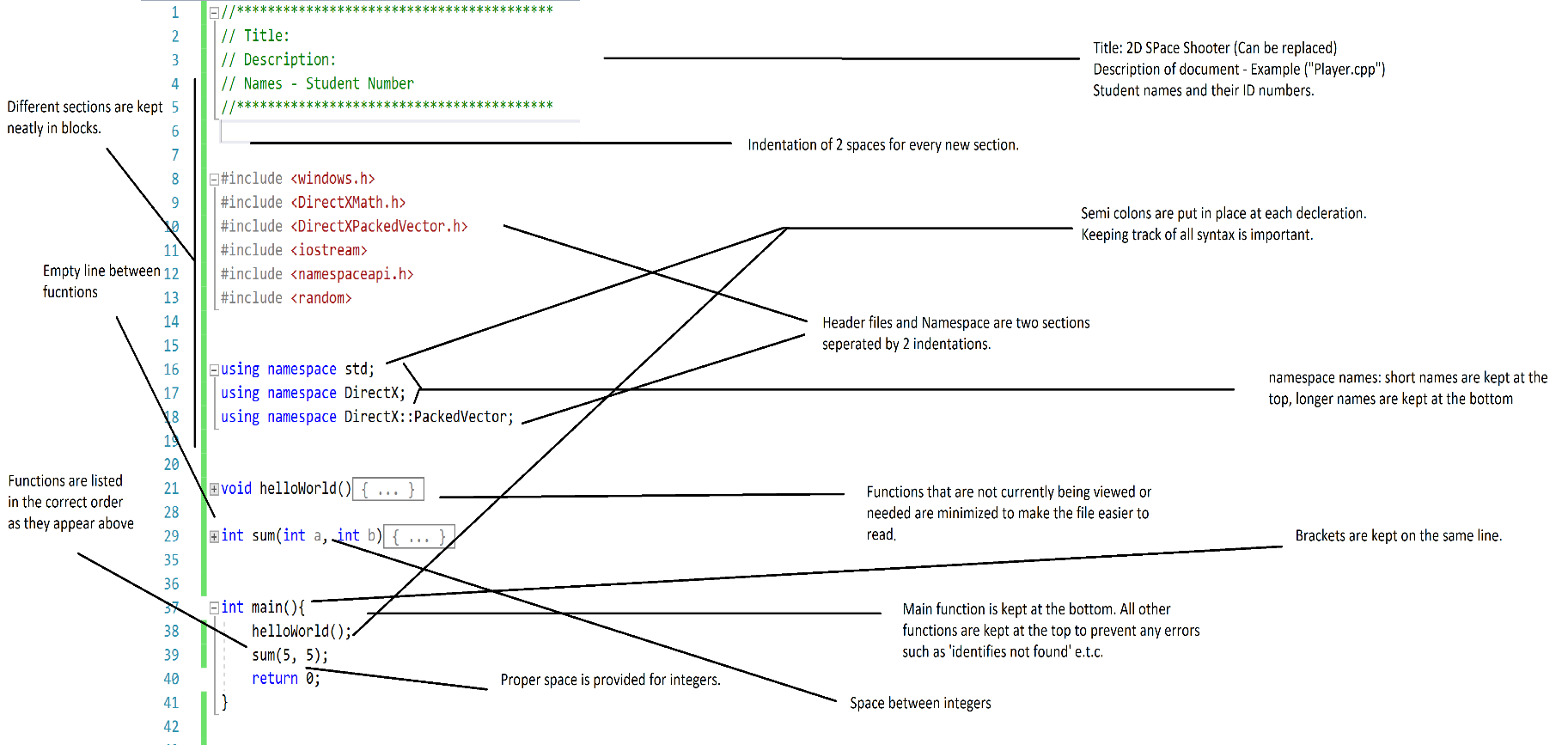
- Milestone 2

- Etc.

**Part 2: Game Engine**

**Coding Styles**

Our game will have a very simple coding style. First, we’ll try to keep all separate lines of code in different blocks. Each block will have 2 indentations between them. Headers and namespaces will be separated by two lines. We will never mix those lines of code. Each block will only contain their types. We’ll have a short description on top of every file as seen in this picture (green text). If the file has a lot of code, we’ll minimize all functions except for the ones we currently need. This will make the file easier to read. Each declaration will have semi colons at the end of them – we think keeping track of every syntax is vitally important. All functions will start with a lower case. Comments will be kept as short as possible. No longer than 1 sentence.



**Project structure**

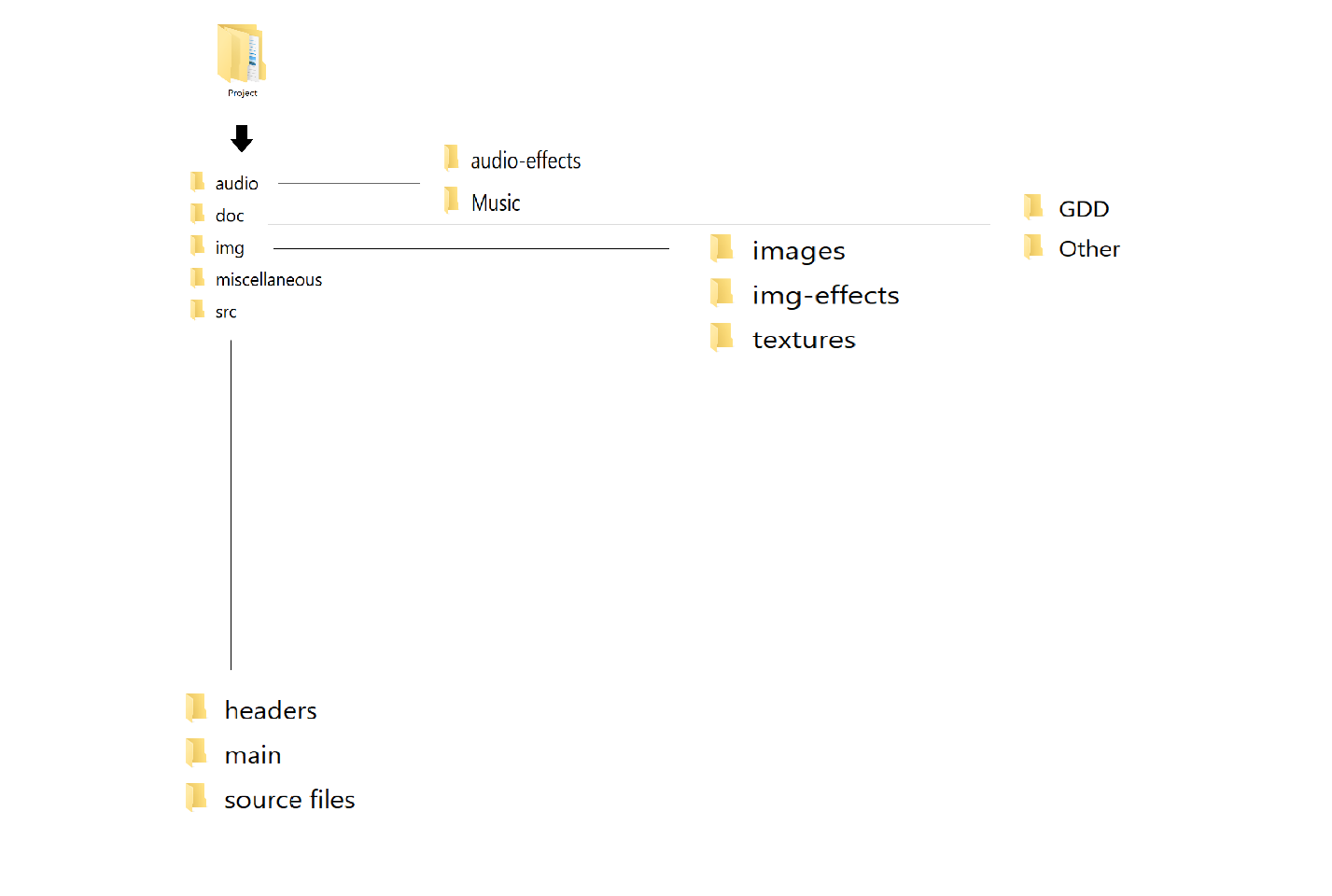
Our project will be very neatly organized. We’re trying to name our folders accordingly. Our main directory folder (Project) will start at the top. It will contain audio, doc, img, miscellaneous, and src directory.

The audio directory will contain the audio effects such as lasers shooting and background music, which will change accordingly to each new level.

The documentation directory will contain the GDD and any other necessary documents to provide a description of the game, the engine or other.

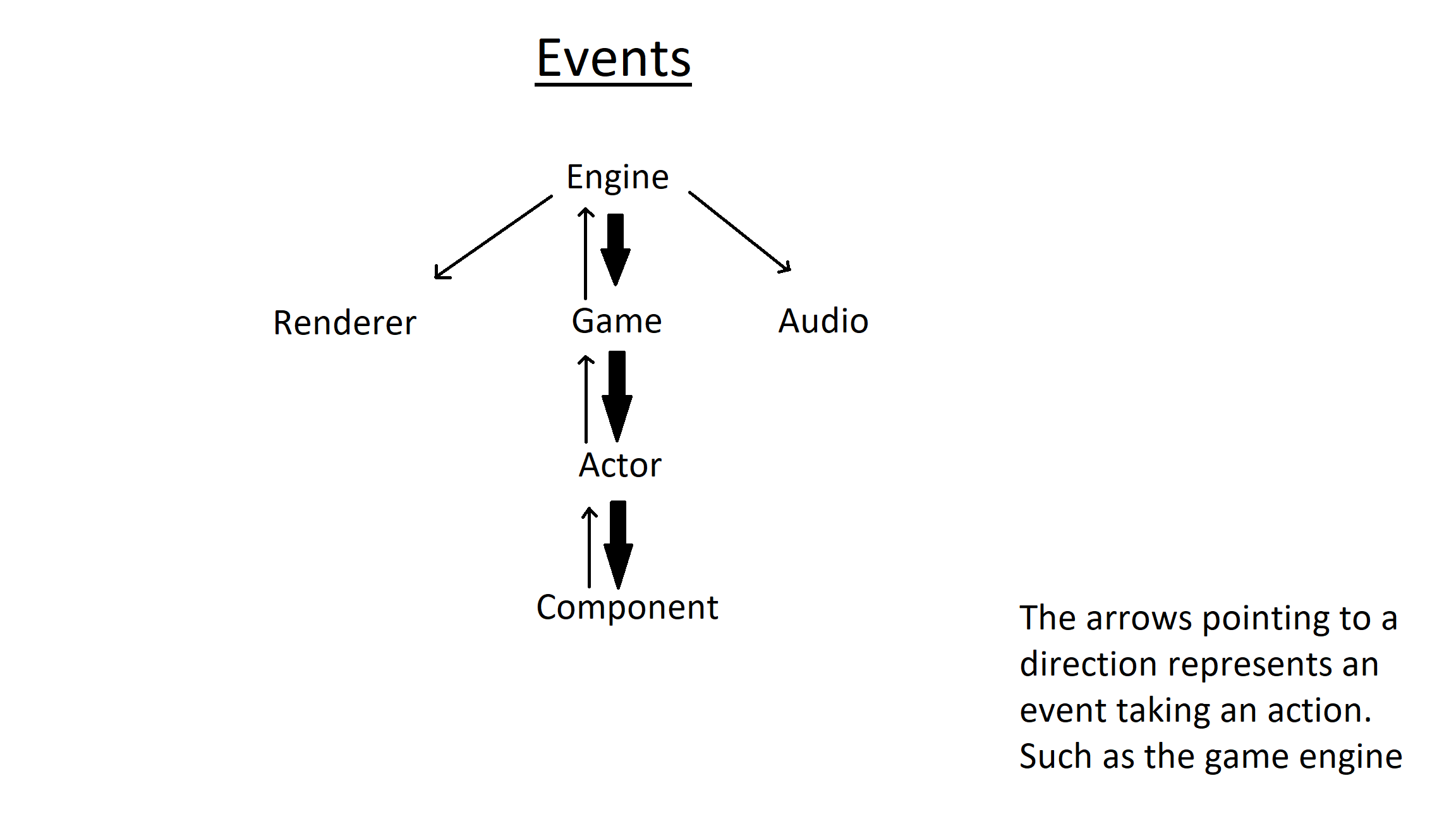
The image directory will contain all the images that you will see in the game. The images folder will contain the background images, img-effects will show off laser images and explosions and the texture directory will give our 2D spaceship some color. Enemies will also be given textures and buildings too.

The miscellaneous directory will contain all other types of files that we are unable to organize efficiently.  
  
The source (src) directory will contain all our sources. This file will contain 3 more files: headers, main, and source files. The headers directory will contain all our header files, the main directory will contain our main source and the source files will contain all of our classes.  
  
The image below illustrators our project structure.



**Minimum PC requirements**  
  
The 2D space shooter game will have the following minimum game requirements:  
  
Operating System: Windows XP  
Processor: 1.6Ghz  
Memory: 512MB RAM  
Graphis: Basic integrated motherboard graphics card  
Storage: 100MB of available space  
Sound Card: Basic integrated motherboard sound card  
Additional notes: These minimum requirements are not very accurate. This game will run on modern computers. Older computers might show problems, especially the ones running Windows 95 or 2000.

**Component Architecture**  
  
The arrows pointing to a word represents an event taking an action. An example would be the game engine (at the very top) ordering the render() function to proceed or the audio() function to create music.



**Components part II**

The player will be the most important component. He or she will be comprised of the following functions: player.cpp, damage.cpp, lives.cpp, ammo.cpp, and health.cpp. The player will be conjoined with the score.cpp function. The main.Score.cpp (as soon on the right) will be the high score.   
  
The enemies will be comprised of a statusCheck.cpp, health.cpp, and damage.cpp. The status checks to see if an enemy has been destroyed. If it has, a score is added to the scoreboard.   
  
After all the actors have been drawn. The images come next. The background is added to the correct level. And all the image effects are added to the player and enemy players.   
  
The audio comes next and adds the background music and all the sound effects to the game.  
This loop continues until it is back at the player component.   
  
Each component is divided into one block. There are 3 blocks in this game loop.  
  
The third block (audio, and images) is connected to Block I and Block II.  
A separate function, Score (High Score) is kept hidden on the side.   
Also a separate level changer function is also kept hidden on the side.   
The two hidden functions come after Block I. Think of it as Block 1.5.