CS499  
Senior capstone project

Dungeon Warrior

Software Design Document

Version 1.4

Randall Rowland

23 August 2016

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 29 June 2016 | 1.0 | Initial version of Dungeon Warrior’s Software Design Document | Randall Rowland |
| 18 July 2016 | 1.1 | Added interface graphic | Randall Rowland |
| 2 August 2016 | 1.2 | Added flow chart graphic and interface operations. | Randall Rowland |
| 23 August 2016 | 1.3 | Added UML and citations | Randall Rowland |
| 23 August 2016 | 1.4 | Swapped Flow Charts | Randall Rowland |
|  |  |  |  |
|  |  |  |  |

C

ontents

[Revision History ii](#_Toc460600651)

[ontents iii](#_Toc460600652)

[1. Introduction 1](#_Toc460600653)

[1.1. Purpose 1](#_Toc460600654)

[1.2. Goals and Objectives 1](#_Toc460600655)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc460600656)

[1.4. References 2](#_Toc460600657)

[1.5. License 2](#_Toc460600658)

[2. Design Overview 4](#_Toc460600659)

[2.1. Introduction 4](#_Toc460600660)

[2.2. Technologies Used 4](#_Toc460600661)

[2.3. System Architecture 4](#_Toc460600662)

[2.4. System Interfaces and Operation 5](#_Toc460600663)

[2.5. Constraints and Assumptions 8](#_Toc460600664)

[3. User Interface Design 9](#_Toc460600665)

[3.1. Description of the User Interface 9](#_Toc460600666)

[3.2. Interface Design Rules 11](#_Toc460600667)

[3.3. Objects and Actions 11](#_Toc460600668)

[4. Class Documentation 13](#_Toc460600669)

# Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including sequence diagrams, collaboration models, object behavior models, and other supporting requirement information.

## Purpose

The purpose of this software design document is to provide a low-level description of Dungeon Warrior, providing insight into the structure and design of each component. Topics covered include the following:

* Class hierarchies and interactions
* Data flow and design
* Processing narratives
* Algorithmic models
* Design constraints and restrictions
* User interface design
* Test cases and expected results

In short, this document is meant to equip the reader with a solid understanding of the inner workings of Dungeon Warrior.

## Goals and Objectives

Dungeon Warrior is a single, comprehensive programming project using the C++ programming language. The goal is a proof of concept program to demonstrate learning objectives learned throughout American Sentinel University’s Bachelor of Science in Computer Science Game Programming Specialization degree program. Dungeon Warrior will incorporate the learning objectives from:

* CS130 – Introduction to Computer Programming
* CS205 – Intermediate Computer Programming
* CS221 – Software Engineering
* GP210 – Introduction to Game Design
* GP221 – Introduction to Game Programming
* GP312 – Computer Graphics Programming
* GP435 – Artificial Intelligence for Gaming

The objectives of Dungeon Warrior:

* Apply software engineering techniques to a larger-scale problem
* Integrate appropriate computer science theory, concepts, and methods
* Demonstrate proper documentation
* Display comprehensive programming knowledge

## Definitions, Acronyms, and Abbreviations

* **ADT –** Abstract Data Type. A collection of data values together with a set of well-specified operations on that data.
* **AI** – Artificial Intelligence.
* **API** – Application Program Interface
* **Collision** – Determining if an object has intersected another object or overlapped relevant background scenery.
* **DirectX** - a collection of APIs for handling tasks related to multimedia, especially game programming and video, on Microsoft platforms.
* **Object** – Is a data structure that has state (data) and behavior (code). Objects correspond to things found in the real world.
* **OOP** – Object Oriented Programming. Programming language model organized around objects rather than “actions” and data rather than logic.
* **OpenGL** - Open Graphics Library (OpenGL) is a cross-language, cross-platform API for rendering 2D and 3D vector graphics.
* **Scholarship** – Academic study or achievement; learning of a high level
* **SDD** – Software Design Document.
* **UML** – Unified Modeling Language. For definition and uses, see <http://www.uml.org/what-is-uml.htm>

## References

Christopho. (2016, January 25). Zelda ALTTP resource pack for Solarus. Retrieved June 29, 2016, from <https://github.com/christopho/solarus-alttp-pack>

Customan. (2012, November 20). Folder Games Icon. Retrieved July 18, 2016, from <http://www.iconarchive.com/show/christmas-icons-by-custo-man/Folder-Games-icon.html>

Morrison, M. (2005). Beginning game programming. Indianapolis, IN: SAMS.

Schell, J. (2008). *The art of game design: A book of lenses*. Amsterdam: Elsevier/Morgan Kaufmann.

## License

### Software Design Document/Source Code License

This document, the icon and the source code for Dungeon Warrior are licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode).



Under this license, you are free to:

* **Share** – copy and redistribute the material in any medium or format
* **Adapt** – remix, transform, and build upon the material

Under the following terms:

* **Attribution** – You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggest the licensor endorses you or your use.
* **NonCommercial** – You may not use the material for commercial purposes.
* **ShareAlike** – If you remix, transform, or build upon the material, you must distribute your contribution under the same license as the original.

### Zelda: A Link to the Past Copyright

The Zelda fonts, music, sounds, sprites, tile sets, and other resources used from Zelda: A Link to the Past are copyrighted and owned by Nintendo.

As Dungeon Warrior is for scholarship, these resources fall under Fair Use. For this particular instance. However, if the project is shared outside of educational purposes or for commercial use, these resources must be removed.

17 U.S. Code § 107 - Limitations on exclusive rights: Fair use

Notwithstanding the provisions of sections 106 and 106A, the fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—

(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;

(2) the nature of the copyrighted work;

(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) the effect of the use upon the potential market for or value of the copyrighted work.

The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

# Design Overview

## Introduction

The Design Overview is section to introduce and give a brief overview of the design. The System Architecture is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient themselves to the design and see a summary before proceeding into the details of the design.

## Technologies Used

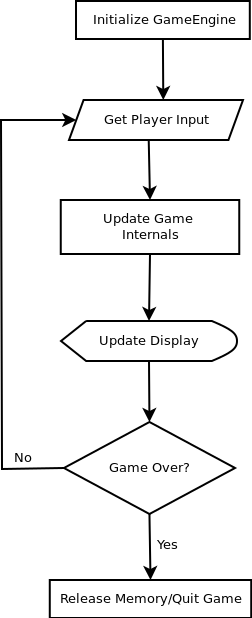
### Hardware

* Hewlett-Packard ProBook 640G1
  + Intel® Core™ i3-4000M
  + 8 GB Ram
  + Windows 7 Enterprise SP1 (64-bit)
* Lenovo X1 Carbon
  + Intel® Core™ i7-5667U
  + 8 GB Ram
  + Ubuntu 16.04 LTS (64-bit)
* Alienware Mx11-R2
  + Intel® Core™ i7
  + 8 GB Ram
  + Windows 7 Home Edition SP1 (64-bit)
* Custom Desktop PC
  + AMD Phenom
  + 16 GB Ram
  + Ubuntu 16.04 LTS (64-bit)

### Software

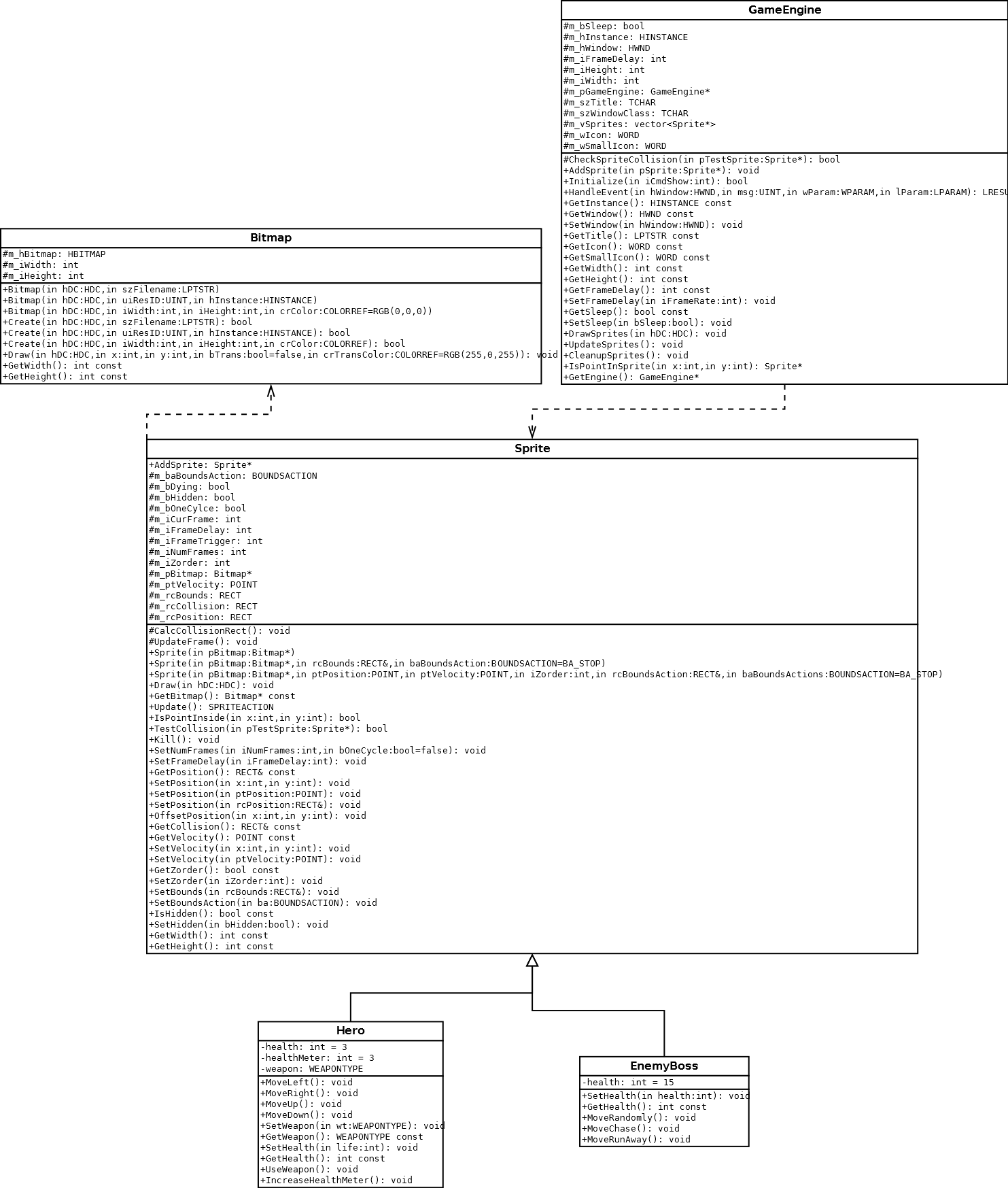
* Microsoft Visual Studio Community 2015 (Version 14.0.25421.03 Update 3)
* Microsoft Visual Studio Code
* Microsoft Office 2016 Professional Plus
* Atlassian SourceTree
* Pinta 1.6
* Microsoft Word 2013
* Syntevo GmbH SmartGit 7.1
* Dia 0.97.3
* Geany 1.27
* Tiled Map Editor 0.16.1

## System Architecture

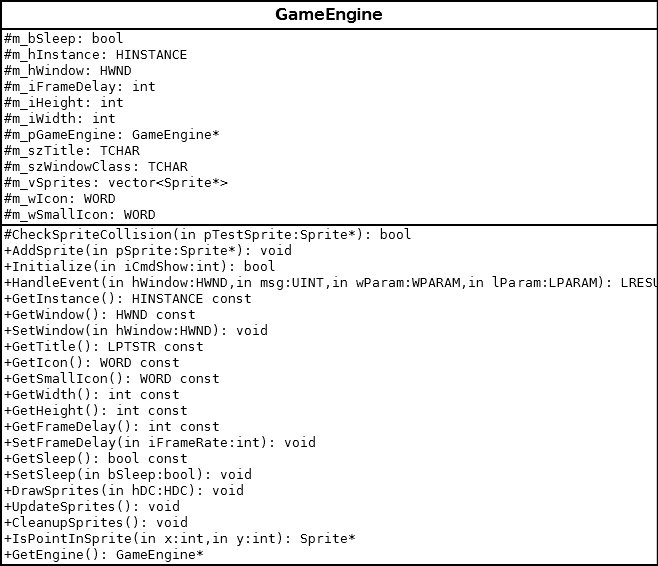


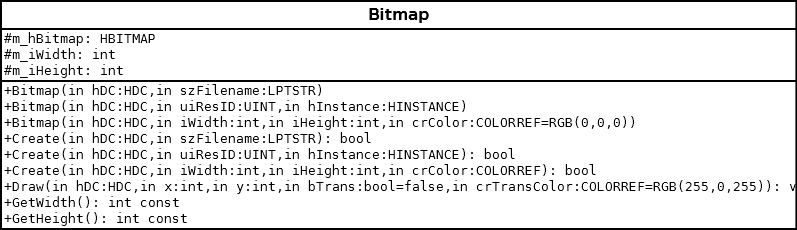
## System Interfaces and Operation

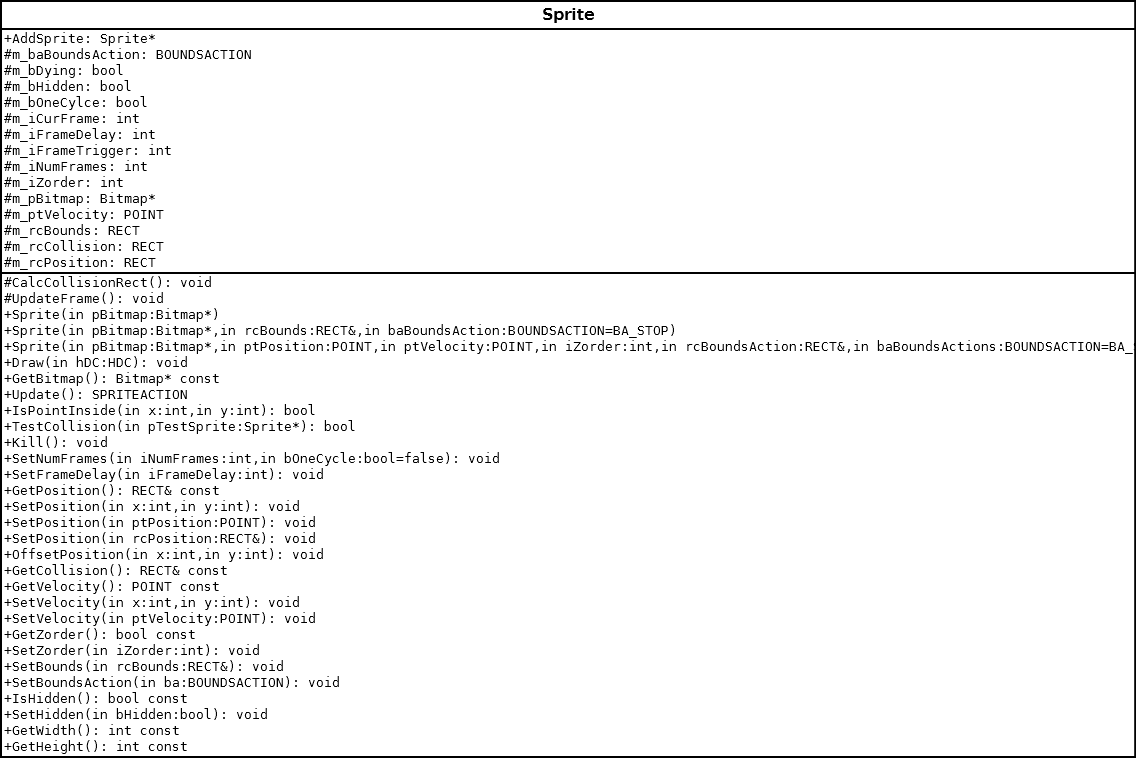
Overview of how the system interacts with one another.

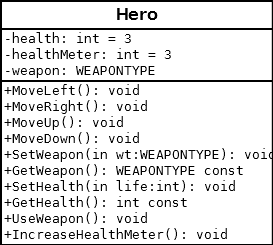
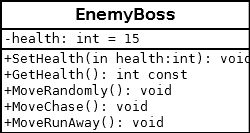


The following is a close up of each class of UML:







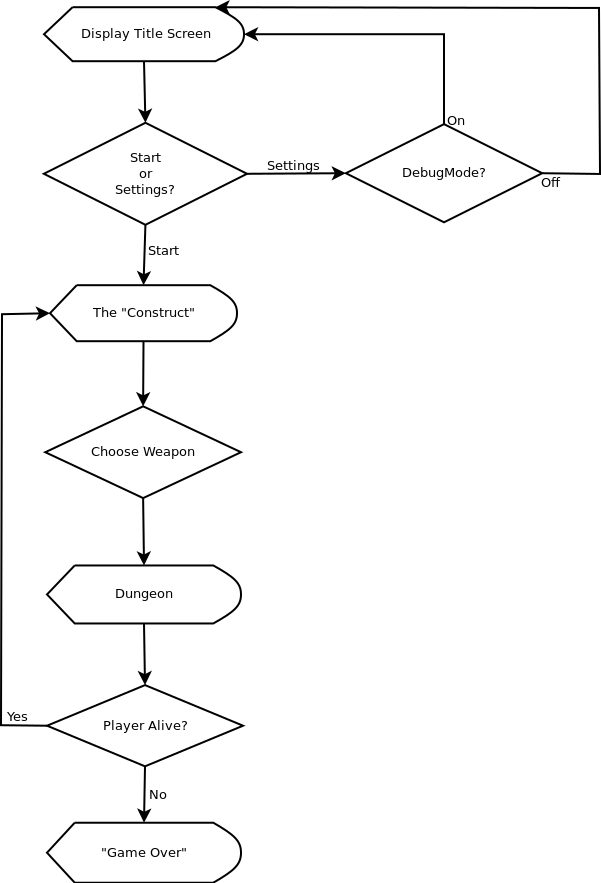
## Constraints and Assumptions

Due to the learning curve and ever changing technology of Microsoft’s DirectX API and OpenGL API, this game was programmed using just Microsoft’s Windows API. Although, DirectX or OpenGL would provide a more robust game, the eight-week time constraint on the project makes using those APIs near impossible. I would also have to test it out on more machines and see which dependencies would have to be included to ensure it runs. The assumption is that by using the Windows API, this game should run on any modern Windows version.

# User Interface Design

## Description of the User Interface

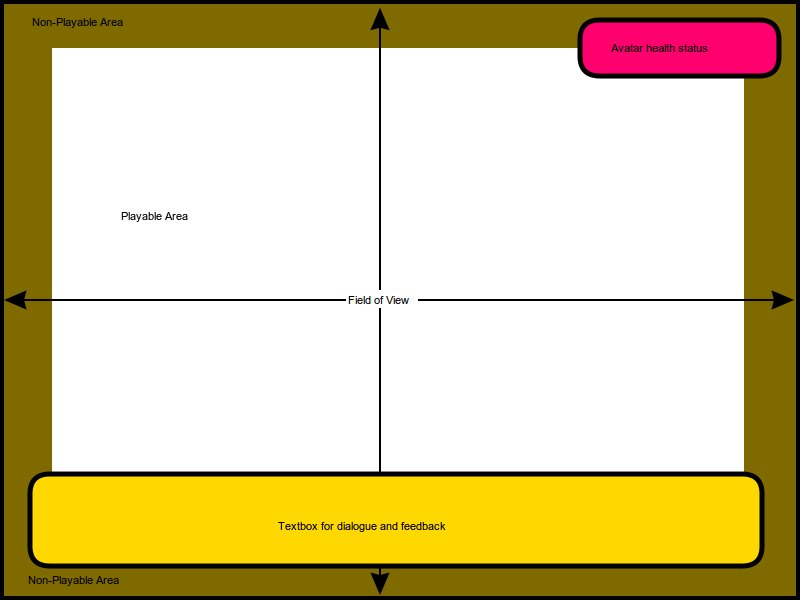
**Title Screen.** The will show a beautiful graphical background with text over the top show casing the title of the game, “Dungeon Warrior“. Towards the bottom of the title screen will be two items that will be selectable by using the arrow keys to highlight one of the items and using ‘Enter’ or ‘Space Bar’ to select it. The two items will be graphical text that states: “Start Game” or “Settings”.  
**The “Construct”.** This is where the game will start off at. The name comes from The Matrix. In The Matrix, the Construct is a virtual workspace that is used as their loading program. They are able to load virtual objects that are then used within The Matrix. This same concept applies here. When the game starts off you will be in an empty room where an old man will greet you. If you know about The Hero’s Journey concept, this old man is the Mentor. He will tell you to choose one of the three weapons in front of you, which will be a bow with arrows, a sword, and a boomerang. Once you select one and confirm your selection, you will be transported to the dungeon. The arrow keys will move the player around and the ‘Spacebar’ will be used to make selections. “This…is the Construct. It’s our loading program. We can load anything, from clothing…to equipment...weapons…training simulations…anything we need.” -Morpheus  
**Dungeon.** Inside the dungeon is where the player will battle enemies using the weapon they chose from The Construct. Once all the enemies are defeated they will be transported back to The Construct to switch weapons if they choose to. Each time the player defeats all the enemies and returns from The Construct, the dungeon will increase difficulty. More enemies or a completely different enemy or a combination of both will happen each time the play enters the dungeon. If the player is unable to defeat the enemies and dies, the game ends and will display the game over screen. The player will use the arrow keys and ‘Spacebar’ on the keyboard to control their avatar. The arrow keys will move the player around and the ‘Spacebar’ will activate the avatar’s weapon.  
**Game Over.** If the player is unable to defeat the enemies in the dungeon and dies. The game over screen will appear to let the player know how many waves of enemy they were able to defeat. From this screen they can press ‘Enter’ or ‘Spacebar’ and the game will release resources from memory and close the program.  
**Settings.** When settings are selected from the title screen, a pop-up balloon style window will appear. At the top of the window will be text that says, ‘DEBUG’. Below that will be text that will say ‘ON’ and ‘OFF’. Default when the program starts will be set to off. No matter which one is selected, the window will close and you will be presented with the Title Screen again. If debug is set to on, the frames per second will be displayed in the corner and boundary boxes around sprites will display so you can see collisions. The player will be able to select on or off using the arrow keys and confirming the selection with the spacebar.



## Interface Design Rules

The interface design rules for Dungeon Warrior are derived from The Art of Game Design. I used the Lens of Simplicity and Transcendence, Lens of The Hero’s Journey, Lens of Status, Lens of Action, Lens of Goals, Lens of Virtual Interface, and Lens of the Avatar.

## Objects and Actions



This is a general layout of how the user interface will look. Proportions may not be exact in the final product. End user will interact with the user interface and avatar using the arrow keys and space bar on the keyboard. **Field of View:** Inside of the Field of View will be a set size and the end user will not be able to change the size to ensure correct aspect ratio and view. Field of View will be contained within the Window and will not scroll.  
**Non-Playable Area:** Inside of that window will be a border. Border will use wall type graphic to give the illusion the avatar is in a room. This is depicted above with the brown/bronze color and labeled as a Non-Playable Area. This will stop the avatar from “walking” off the screen. Although depicted all the way to the edge of the Field of View, this is only an example. Some levels may be smaller and have a smaller Playable Area.  
**Playable Area:** The avatar will be able to move anywhere within the white area or Playable area. Border will prevent avatar from accessing non-playable areas. EXCEPTION: Elements may be added to the playable area to give an aesthetic look and challenge the avatar. The avatar may or may not be able to “walk” through those elements.  
**Textbox:** This portion depicted with the yellow/gold box above will not be visible all the time. This will be a popup box when the avatar is interacting with other entities. When this popup box appears, all elements within the Playable Area will pause. The main focus will be on the textbox and the end user will only be able to interact with the textbox until it is complete and goes away. Then the Playable Area will resume normal game play.  
**Avatar Health Box:** The pink box depicted in the top right corner will be sometime of floating health meter to provide the avatars health status to the end user. This will be visible at all times during game play.

# Class Documentation

## Bitmap Class Reference

### Detailed Description

Takes a bitmap and turns it into a Bitmap object that will be drawn to the screen.

There are four different types of constructors used to create a Bitmap object. You can draw the bitmap with or

without transparency. Also allows only a part of a Bitmap to be drawn. This can be use full if you have 'frames' that could allow animation or if you want to just use a tilesheet for several images.

**Public Member Functions**

• Bitmap ()  
 Empty constructor that sets width and height to 0, and handle to NULL.  
• Bitmap (HDC, LPTSTR)  
• Bitmap (HDC, UINT, HINSTANCE)  
• Bitmap (HDC, int, int, COLORREF crColor=RGB(0, 0, 0))  
• virtual ~Bitmap ()  
 A destructor that calls the Free() helper function to delete the object.  
• bool Create (HDC, LPTSTR)  
• bool Create (HDC, UINT, HINSTANCE)  
• bool Create (HDC, int, int, COLORREF)  
• void Draw (HDC, int, int, bool bTrans=false, COLORREF crTransColor=RGB(255, 0, 255))  
• int GetWidth () const  
 Returns the width of the bitmap.  
• int GetHeight () const  
 Returns the height of the bitmap.

• void DrawPart (HDC, int, int, int, int, int, int, BOOL, COLORREF)

**Protected Member Functions**

• void Free ()

A helper fuction that deletes the Bitmap object and releases the handle from memory