Software Design Document

for

<Project>

Team: Bravens

Project: The Mono Game

Team Members:   
*Casey Martin*

*Joan Njenga*

*[Insert Member 2 Name]*

*[Insert Member 3 Name]*

*[etc.]*

Last Updated: [3/13/2025 9:21:44 PM]

# Table of Contents

[Update the Table of Contents]

[Table of Contents 3](#_Toc351182317)

[Document Revision History 4](#_Toc351182318)

[List of Figures 5](#_Toc351182319)

[List of Tables 6](#_Toc351182320)

[1. Introduction 7](#_Toc351182321)

[1.1 Purpose of the System [not required] 7](#_Toc351182322)

[1.2 Architectural Design Goals 7](#_Toc351182323)

[1.3 Scope 7](#_Toc351182324)

[1.4 Definitions and References [not required] 7](#_Toc351182325)

[2. Software Architecture 8](#_Toc351182326)

[2.1 Overview 8](#_Toc351182327)

[2.2 Subsystem Decomposition 8](#_Toc351182328)

[2.3 Hardware/Software Mapping 8](#_Toc351182329)

[2.4 Persistent Data Storage and Management 9](#_Toc351182330)

[2.5 Access Control and Security [not required] 9](#_Toc351182331)

[2.6 Global Software Control [not required] 9](#_Toc351182332)

[2.7 Boundary Conditions [not required] 9](#_Toc351182333)

[3. Subsystem Services 10](#_Toc351182334)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 0.0 | 3/10/2025 | Initial Draft | Created based on project overview |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Figures

Figure # Title Page #

S-x Component Diagram #

S-y Sequence diagram for AI Spawns Enemies #

S-y Sequence diagram for user control players #

S-y Sequence diagram for Players Hit by Enemy Bull #

(etc.)

**All figures/diagrams/images included in your document must be numbered and titled. This list catalogues all such items.**

**Figure numbering instructions:**

* **Each figure number will identify the section in which the figure appears and the sequential number of the figure in that section, separated by a hyphen. For example, the first figure in Section 1 Product Concept will be labeled “Figure 1-1”, and the second figure in that section, if applicable, will be labeled “Figure 1-2”. Likewise, the first figure in Section 3 Customer Requirements would be labeled “Figure 3-1”.**
* **The title given each figure should be concise and descriptive.**

# List of Tables

Figure # Title Page #

1 Subsystem overview #

2 Design Patterns Used #

(etc.)

**All ancillary tables/lists included in your document, with the exception of the Table of Contents, Revision History, List of Figures, and this table must be numbered and titled. This list catalogues all such items.**

**Table numbering instructions:**

* **Each table number will identify the section in which it appears and the sequential number of the table in that section, separated by a hyphen. For example, the first Table in Section 1 Product Concept will be labeled “Table 1-1”, and the second figure in that section, if applicable, will be labeled “Table 1-2”. Likewise, the first figure in Section 3 Customer Requirements would be labeled “Table 3-1”.**
* **The title given each table should be concise and descriptive.**

# 1. Introduction

The Bravens project is a bullet hell shooting game developed using the MonoGame framework. The game features a component-based architecture that promotes modularity and reusability, allowing for flexible game object behaviors. The project includes various assets, game logic, and manager classes to handle different functionalities, such as enemy waves and player controls.

## Purpose of the System [not required]

The purpose of the "Bravens" system is to provide an engaging bullet hell shooting experience where players can control a character, fire projectiles, and defeat waves of enemies and bosses.

## Architectural Design Goals

This section identifies qualities that the system should focus on, including:

* **Performance**: The game should run smoothly with multiple enemies and projectiles on screen. Tactics to promote this quality include optimizing rendering and using object pooling for projectiles.
* **Modularity**: The component-based architecture allows for easy addition and modification of game behaviors without affecting other parts of the system. Tactics include encapsulation and composition.

## Scope [not required]

* *Section 2 includes the software architecture of the system.*
* *Section 3 includes the services provided by each subsystem.*

## Definitions and References [not required]

* **Hitbox**: A defined area for collision detection.
* **GameObject**: An entity in the game that can have components attached to it.

# **2. Software Architecture**

This section documents the architectural model of the Bravens’ implementation of the Mono game.

## [For 2.1, follow these instructions:

## Consider these 3 key use cases for our project: “AI spawns enemies”, “User controls player to move”, and “Player is hit by enemy bullets”

## Draw a sequence diagram for each of the 3 use cases as. Include your sequence diagrams HERE, above section 2.1.

## Follow either Multi-layered or MVC architectural pattern (if you want to use others, discuss with the instructor first) to model the overall architecture of your system, and place the component diagram of your architecture in 2.1. For each subsystem draw a class diagram, with as many details as you can.]

## 2.1 Overview

[The architecture follows a component-based design pattern, allowing for flexible and reusable components that can be attached to game objects. The main subsystems include the GameCore, GameObjectManager, WaveManager, and various enemy and player components. Below is the component diagram illustrating the architectural pattern adopted.

*[Insert Component Diagram Here]*

*Introduce the architectural pattern that you use in your design model, and state the rationale in choosing that pattern. Briefly describe the overall decomposition of your system in terms of subsystems.*

*Provide a UML diagram (component diagram or class diagram) that shows your subsystem decomposition based on the architectural pattern that you selected (either Multi-layered, or MVC architecture) The UML component diagram should illustrate the architectural pattern that you adopted. Clearly identify the different subsystems, which will be further elaborated in Section 2.2.]*

## 2.2 Subsystem Decomposition

[In this subsection, describe how you decompose your system into subsystems. For each subsystem you included in Section 2.1, include a subsection with its name, and provide its brief description of said subsection. In the end, add another subsection 2.2.X titled “Design Patterns”. In this subsection, write down the names of at least three different Design Patterns that you have used in your code. For each pattern, also draw the class diagrams from your code that shows how they follow the structure of the patterns.

\*Note: for this part, your diagram should be based on your actual code, to show that you are actually following your design. You only need to include class information (attributes & operations) that are relevant to the pattern. For example, if you follow a Creational Pattern to create new products, you do not need to include all information of the products classes, only the significant ones.

* **GameCore**: Manages the overall game loop and initializes game objects.
* **GameObjectManager**: Responsible for creating, updating, and rendering game objects.
* **WaveManager**: Manages the spawning of enemy waves and bosses.
* **PlayerControls**: Handles player input and movement.
* **PlayerGun**: Manages the firing of player projectiles.

**Design Patterns**:

1. **Component Pattern**: Used to allow flexible attachment of behaviors to game objects.
2. **Singleton Pattern**: Used in GameObjectManager to ensure a single instance manages all game objects.
3. **Observer Pattern**: Used for event handling, where game objects can subscribe to events like collisions.

[Insert Class Diagrams for Each Pattern Here]

In addition, if you feel your design does not follow some pattern “exactly”, that’s OK too. Explain which pattern you think your designs resemble the most and what are the differences. ]

## 2.3 Hardware/Software Mapping [not required]

[Describe how the subsystems are assigned to hardware and off-the-shelf components.

1. Explain the allocation of subsystems to computers and the design of the infrastructure for supporting communication between subsystems. [optional]
2. State the operating systems that run on the computers and any software component that are needed.
3. Provide the UML deployment diagram that shows the allocation of subsystems to devices and execution environments. The nodes of the UML diagram will be the computers that run the subsystems (see Figure 7-21 in the textbook). [optional]

## 2.4 Persistent Data Storage and Management [not required]

[Describe the persistent data stored by the system and the data storage and management strategy used to store and manage those data.

1. Identify the persistent data that need to be stored.
2. Select a storage strategy. Briefly explain why you chose that storage strategy. Discuss pros and cons.]

## 2.4 Access Control and Security [not required]

[Describe the user model of the system in terms of an access matrix. Also describe the security issues, selection of authentication, the use of encryption, and the management of keys.]

## 2.4 Global Software Control [not required]

[Briefly describe how the global software control is implemented.

1. How requests are initiated?
2. How subsystems synchronize?]

## 2.4 Boundary Conditions [not required]

[Describe the start-up, shut-down and error behavior of the system.]

# 

# **3. Subsystem Services**

*[Briefly describe the services provided by each subsystem by writing down a* ***list of interfaces and what their purposes/functions are in your system in this section.***

This section describes the services provided by each subsystem.

* **GameCore**:
  + **InitializeGame**: Initializes the game and starts the main loop.
  + **UpdateGame**: Updates the game state each frame.
* **GameObjectManager**:
  + **AddObject**: Adds a new game object to the scene.
  + **RemoveObject**: Removes a game object from the scene.
* **WaveManager**:
  + **SpawnWave**: Spawns a new wave of enemies based on predefined logic.

[Insert UML Component Diagram Here]

*First, identify the dependencies among subsystems, and explain their responsibilities. Then define the services that each subsystem should provide to and/or require from other subsystems. Give a unique name to each service and explain it in 1 or 2 sentences.*

*Provide a UML component diagram where services are shown with ball-and-socket notations. For simplicity you may include only the subsystems that have dependencies with each other. If your component diagram in 2.1 includes all the names of the services, you may simply refer to that diagram. Otherwise, include a new diagram that highlights these services.*

*Remember, in architecture, services refer to the* ***interfaces among different subsystems, not within.*** *DO NOT describe the services by specifying internal details of the subsystems, which should be done in Section 2.2]*