Contra Development

CourseID: CS 360 - 001

Milestone # and Date: Milestone - 9/31/21

Project Members: Ethan Moore, Aman Patel, Will Craddock

Client: Dr. Galloway

[**Introduction**](#_2iubex965bh5) **1**

[**Project Scope**](#_hwj1feww8h6) **4**

[**System Boundaries**](#_khp32ez5kyhn) **5**

[**Nonfunctional requirements**](#_s69um3n7krky) **6**

[**Functional Requirements**](#_rlrm75y37l43) **6**

[**Structural UML Diagrams**](#_jdc14eo5dnmw) **6**

[Use cases diagram](#_n4vngk1zqvlj) 6

[State Diagram](#_m37279tsp1k5) 10

[Sequence Diagram](#_20re5ju54kvt) 11

[**Organizational approach**](#_rev79beezktx) **11**

[**Technical feasibility**](#_tb2t5jgabcvk) **12**

[**Schedule feasibility**](#_15ofalxy3ej4) **12**

[**Progress visibility**](#_lopa408ioysy) **14**

[**Risk analysis.**](#_51e67fo6s4cj) **15**

[**Wireframe diagrams/Storyboards**](#_6tu2n7y89j0z) **16**

[**OO Design Patterns**](#_gywihfvvwhn8) **18**

[Creational - Abstract Factory](#_cae64goo7w7p) 18

[Structural - Facade](#_1xc6asrchb4v) 19

[Behavioral - Observer](#_mvca5l6rb1nw) 20

[**Structural UML Diagram**](#_e8a00xin3lz) **21**

[Class Diagram](#_4cungd7rsf8s) 21

[Component diagram](#_wdogss2bmtn0) 22

[Deployment diagram](#_r06zs9an6bwt) 23

# 

| Contra Player Select Screen |
| --- |

# 

# Introduction

Contra is a run and gun game originally created for arcades and other publicly available places. Original publisher of the game was Konami, and the game was published on February 20, 1987. And in 1988 home version was released for Nintendo Entertainment System, along with various computer formats. Different regions were calling this game with different names, for example the United States was calling it as Contra, PAL region or European were calling it Gryzor or probotector. The Japanese were calling it Kontora. And after the first success in 1988 they published several sequels of Contra since the original game, but none was more popular like the original game.

Let’s talk about the different characters in the game, contra had a lot of characters from different monsters to game heroes and aliens and more. First two are the face of the game, Bill Rizer and second is Lance Bean, they are future Earth’s premier alien-skull-busting specialists. They first appeared in the arcade version of the game and people loved them so very much they have been the default characters ever since. Bill Rizer’s character design is meant to evoke actor Arnold Schwarzenegger. And his name is a combination of two famous actors Bill Paxton (Bill) and Paul Reiser (Rizer). They both appeared in the action Alien movie when the game was first launched. Bill Rizer has then appeared in Contra Return, Super Bomberman R, Contra Evolution, Konami Classics Vol.1, Contra ReBirth, Contra 4, Konami Classics Series: Arcade Hits, Neo Contra, Konami Collector’s Series: Castlevania & Contra and many others.

Second player character Lance Bean is meant to evoke actor Sylvester Stallone’s John Rambo character. And his name is the combination of two actors: Lance Henrikssen (Lance) and Michael Biehn (Bean). Both appeared in the Alien action movie. And he was appeared in the games like Contra Return, Super Bomberman R, Contra Evolution, Konami Classics Vol.1, Contra ReBirth, Contra 4, Konami Classics Series: Arcade Hits, Neo Contra, Konami Collector’s Series: Castlevania & Contra and may other

There was a third character called Lance Gryzor which was present only in the European version of the game and only appeared in the first version of the game. And this version of the character was the combination of both Lance and Bill.

Now we will talk about the popular monsters of the contra series. One of the most dangerous and famous of all was the Java monster also known as Red Falcon of Red Falcon army and one of the highest ranked members of the army too. He was the highest-ranking leader in the command of the first alien invasion attempt on Earth in 2633 A.D. He is recognized as two different monsters, his upper body and head are known as “Emperor Demon Dragon God Java”, While his heart is known as “Gomeramos King”. His appearance in the game is just like his background stories, Java vomits an endless supply of the Bundles(flying and persistent grub-like creatures that attack the player). The player is continuously given new bundles by the Java and the player cannot move to the next corridor until the Java is destroyed.

Once the player has destroyed the upper body and the head of the Java monster his heart remains functioning and is characterized for covering up an entire wall, pulsating rapidly as it provides life to the rest of the biomass. And this monster is known as “Gomeramos King”. And it is protected by the alien eggs surrounding it on both ground and ceiling level against potential threats, which release an unrelenting hoard of scorpion- like creatures. Java first appeared in Contra III: The Alien Wars, as mini-bosses that are fought throughout the final stage.

Another famous weapon/boss of the game is Dogra which is nothing but a tank that shoots cannons and has a spike in the front bumper. Dogra has appeared twice in the stage number 5 known as “Snow Field”. And as the stages are timed it is important to do more damage in less time. So, weapons like Spread Gun, Machine Gun or Rapid-Fire Laser are really helpful in successfully destroying this vehicle. Dogra changes the colors as it takes the damage from the player, and it proceeds to aim the cannon at the player’s current position. In different versions of the game the destruction of the Dogra is meant to be different. For example, in Contra: Evolution Dogra’s can be destroyed with a single stab from Sally’s katana.

Defense Wall is another famous monster of the. It appears in the very first stage of the game. This stage is called “Jungle” and Defense Wall is the final enemy to proceed to the next stage which is called “Base”. It looks like a tall and heavily armed defense mechanism that usually guards the entrance to an underground enemy base. And after its debut in the original version this wall has been used in every single contra game as a mini- boss or a boss in one of the earlier stages. Until Contra 4 the core remained passive during encounters, but after Contra 4 it was given the ability to project a powerful energy beam. Another Defense wall which is famous for four different power sources is known as “Garmakilma” which appeared at the end of stage 2 called “Base” guarding the stage 3 called “Waterfall”. Here is the table of the different monsters in the different stages of the contra game line.

| **Stage** | **Name** | **Boss** | **Music** |
| --- | --- | --- | --- |
| **1** | Jungle | Defense Wall | Dense Forest Battle |
| **2** | Base | Garmakilma | Labyrinth Fortress |
| **3** | Waterfall | Gromaides | Waterfall of Bloodshed |
| **4** | [Base 2](https://contra.fandom.com/wiki/Contra_Stage_4) | Godomuga | Labyrinth Fortress |
| **5** | Snow Field | Dogra ×2 Guldaf | Fortress in the Ice |
| **6** | Energy Zone | Gordea | Fortress in the Fire |
| **7** | Hangar | Final Gate | Dense Forest Battle |
| **8** | Alien's Lair | Emperor Demon Dragon God Java  [Gomeramos King](https://contra.fandom.com/wiki/Gomeramos_King) | Horrible Heartbeat |

| Table listing Contra level numbers, names, bosses, and music. |
| --- |

Contra is one of the most famous run and gun games of its time. And features like selecting your weapons and characters that evoke actors like Arnold Schwarzenegger and Sylvester Stallone were adding more and more popularity among adults and kids playing games those days. Even today there are a lot of contra fans all around the world playing the desktop version and our team is one of those. Now as we discussed earlier the game has many different weapons, characters, stages and monsters, but in our version we will be not adding all of those monsters and characters due to the shortage of time and resources, in next few pages we will talk about exactly what stages and monsters we will offer in our version.

# Project Scope

As we mentioned earlier, original Contra was offered as an arcade, then there was the Nintendo version and recently there has been desktop and even smartphone versions of the game, but we will just be focusing on the desktop version as discussed with our client Dr. Galloway. And also we will be creating the version of Contra with fewer stages but they will be more interesting and take longer to finish as we intend to have more than one major monster and bosses on one stage. That way we have fewer stages but the same amount of entertainment for the players.

We will be implementing three stages which will be inspired from stages 1(Jungle), 5(Snow Field), and 8(Alien's Lair). But our goal is to add all the monsters and bosses we could to these three stages and also give as much ammunition and weapons we can get to our characters. We will add both main characters to the game. And our game will be able to connect to the gamepad to keep the nostalgia of the original game while also implementing remapping support.

Hardware

The hardware we expect our version of Contra to run on is a Desktop or Laptop. Unity’s requirements for the operating system are either windows 7 or 10. The CPU requirements include x86/x64 architecture with SSE2 support. The graphics API DX 10, DX 11, DX12 capable. Some additional prerequisites are the drivers like visual studio 2015 with the C+ Tools component and Windows 10 SDK. The system resources required to play our remake shouldn’t be intense. The CPU required should be within the last decade. RAM usage should not be greater than 50 megabytes since our game has a more limited scope than the original. The space required to download our remake should be less than a gigabyte since we are not recreating the full game. We are also using the original sprites which are fairly lightweight and bloat free. Our development machines are running modern graphics and processing units. An extra piece of hardware the user may opt for is a NES gamepad. We will develop our game with the Mekela 5.8 foot classic usb wired controller in use.

Software

### 

Front-end frameworks: To be able to create our game and see our behaviors and assets we will use the Unity game engine. The front-end framework’s version we are using is Unity version 2020.3.17f1 LTS. We went with this unity version because it was recommended and the 2021 version could be too new while the older version could be more stable. The scripting language we will use in Unity is C# version 8.0 that uses the Roslyn C# compiler.

Back-end frameworks: To retrieve the saved data from the user we will utilize a back-end framework. The back-end framework we will use for storing the high score, current level, current players items is php my admin and my sql. The database will be locally hosted on the user's computer on an open non conflicting arbitrary port. We won’t use a public facing server to access the database but a locally hosted one for save game data.

# System Boundaries

Physical

The Physical boundaries for this system will be as follows:

-The Game will be run on Unity which wich will be run on windows 10

- We will be using SQLite which will be implemented through the save feature of the game

Logical

The Logical boundaries for this system will be as follows:

* The game will be interacting with the Database for saving user info and saving the state of the the game while in a level
* The player will interact with the system through a computer's keyboard or a NES controller with an USB plug in.

# Nonfunctional requirements

1. The project should be implemented using the Unity game engine and development environment.
2. The project team should create a complete replica of the game.
3. The project can use pre-developed images, artwork, audio, and other related media.
4. All project source code must be developed by the CS 360 project team.
5. The project must use a database.
6. Performance metrics should be gathered and optimized.
7. Security metrics should be gathered and optimized.
8. User interface metrics should be gathered and optimized.

# Functional Requirements

1. Controlled by the keyboard or gamepad.
2. Multiple players to create separate accounts.
3. Login system for users
4. Login system for administrators.
5. Save the state/progress of all players and
6. Allow players to continue once they've logged into their account.
7. Splash screen when launched which also shows top scores and the login prompt.

# Structural UML Diagrams

## Use cases diagram

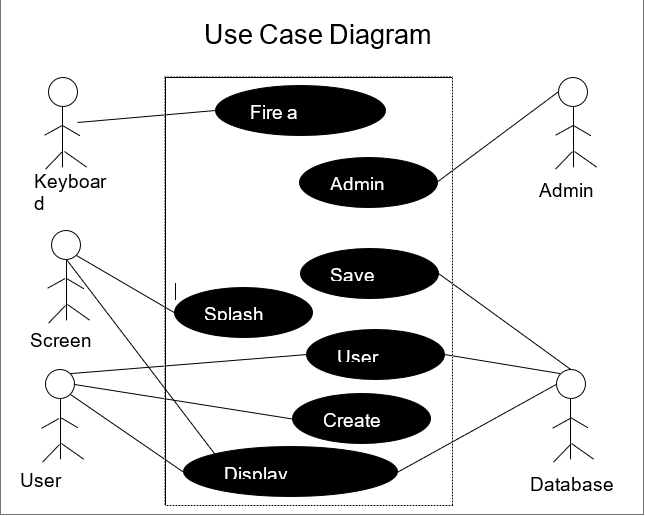
Use case diagrams provide us with a better understanding of how system actors are intersecting with each other. On top of that they also give us a better understanding of system boundaries. Use case diagram consists of three elements. Actors, Use cases and System boundaries.

Use cases: Actions that your system can perform by interacting with internal components and outside actors. In the diagram below we have chosen some of the many use cases possible from the game. Firing a weapon, admin/ user login, saving the progress, creating a new user etc.

System boundaries:Defines the system in relation to external components.In our game there are going to be boundaries as discussed earlier.

Actors:Individuals or external components that interact with your system. In our case these will be users, admins, database etc.

As shown below we have six use cases and five actors interacting with each other to work accordingly. Everything in the big rectangle is in the system boundary and the system is supposed to do when it's functional. According to diagram, user uses keyboard to fire a weapon, Splash screen with the login prompt will be displayed on the screen , user will be able to log in, create new user save their progress, and admin will able to log into the administrator account.



Developed Use case Scenarios

Use case diagrams help us determine boundaries of the system and also get a better picture of how all of the stakeholders work together to make the system functional. But that is not enough to understand the system and why we have the functionality that we have. To tackle that problem we will give examples of developed scenarios with the help of Use cases in the use case diagram and then trace requirements in the requirement traceability table to see if all of the requirements are met.

Use case scenarios are just the broad explanation of use cases in the Use case diagrams. Every scenario has a primary scenario, actors, preconditions and a secondary scenario. Primary scenario is the main explanation of what's going on, preconditions are the requirements and secondary scenarios are the errors, exceptions or branches of the primary scenario. Actors can be the users, admin or database people or the external systems involved in the scenario.

We have two different scenarios here and the ID for both is UC1 and UC2, UC1 has three actors; Player1, Player2, and database. And the precondition is that player1 already exists, player1 starts the game, splash screen is displayed with the login prompt and high score is displayed on the top left corner, once the player logs in the topscore of the player is also displayed below the top score. Now the player starts playing the game using the controls on the keyboard. Now another player wants to join the game but that user does not have the account so he creates a new account and now both players start a multiplayer game. Both of them after playing for a while stop the game and log out of the game and all the progress is saved under their respective accounts. Player1 logs back in after a while and starts playing the game from where he/she left before. And we have one secondary scenario here that is if a user logs in with the wrong password or username it gives a login error.

| Use Case: Contra Game |
| --- |
| ID: UC1 |
| Actors: Player1, Player2, Database |
| Preconditions: Player1 exists |
| Primary Scenario:   1. Splash Screen is displayed with the prompt to Login. 2. Player logs in to the game 3. Top score is displayed 4. The player's top score and level the player is on. 5. Player starts playing the game using the keyboard 6. Another player joins the game. 7. Creates a new account. 8. And both players are playing a multiplayer game. 9. Both users stops playing the game 10. Their progress is saved in the database 11. Both of them log out. 12. First player logs back in and starts the game from where he left before. |
| Secondary Scenario:  Error Logging in |

For the second use case, preconditions are Player1 and Player2 both exist, and actors for this scenario are Admin and Database. Here the admin is logged to his/her account and then deletes a user which has been inactive for a while to free up space in the database. And same as earlier the secondary scenario is the invalid login information.

| Use Case: Contra Game |
| --- |
| ID: UC2 |
| Actors: Admin, Database |
| Preconditions: Player1 and Player2 Exists |
| Primary Scenario:   1. Administrator login to his account 2. Sees what all the players progress 3. Sees Player 2 is not active for a while 4. Deletes Player 2 5. Logs out of the account |
| Secondary Scenario:  Error Logging in  Invalid User |

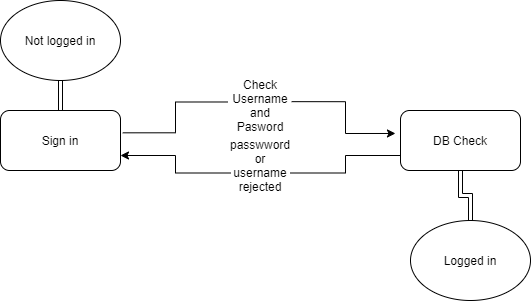
Requirement Tracing

Now we will map the requirement traceability table. Here the requirements are referred to Functional requirements discussed earlier. So according to the table Scenario 1 is using requirements like creating the user, controlling the character using the keyboard, saving the progress, user log in, splash screen, and multiple players. And Scenario 2 is using the admin login requirement.

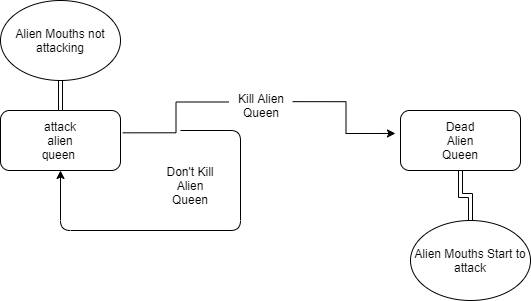
| Requirements | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| UC1 | ✔ | ✔ | ✔ |  | ✔ | ✔ | ✔ |
| UC2 |  |  |  | ✔ |  |  |  |

## State Diagram

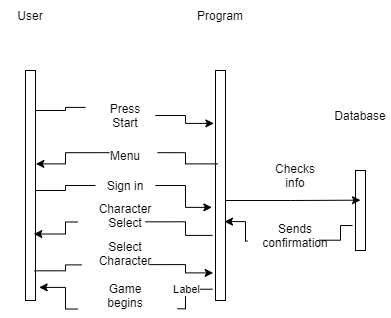
User is not logged in but the state changes to signed in when the database check is successful.



In the final boss the state of the alien queen is determined by the player killing it. If the queen dies then the alien mouth attacks the player.



## Sequence Diagram



Our sequence diagram shows the interaction between the use and program. Some interactions include starting the game by pressing the start key, signing in, or seeing the menu generated by the program. The program interacts with the database by checking information and sending confirmation. The user does not interact directly with the database but uses the program as a proxy.

# Organizational approach

How/when did the group meet: Group has decided to meet twice a week that is Tuesday over the discord call and Thursday in person to discuss the progress and task each group member has to perform. Group also meets with the client every week on Friday at 4:00PM over the Zoom to share the progress of the project. Group has also decided to meet outside of the scheduled time if necessary. The organization and version control method we will use for our project is github. The group member hosting the repository is Ethan Moore or also known as the github user eldm-ethanmoore. The project is the repository Contra-Remake-CS360. This repository will contain our draft, weekly progress reports, and general project files like scripts and assets.

Sprint 1 overall team evaluation: Our team has three members and here is the background of all three members.

* Aman Patel: I am a CS major currently in my senior year, have a little experience in java coding and data structures. Am also familiar with the database functionality. Have worked in groups in the past so I will be an asset for this team moving forward.
* Ethan Moore: My major is also CS and I am in my sophomore year. I have experience with java, python, solidity, some javascript and C/C++. My main interest is in blockchain development and writing smart contracts. The most I have used the unity game engine for is to manipulate the player object via firing a ray at another gameobject to displace myself upward. I have worked on one group project in the past so this will be a new learning experience.
* Will Craddock: I am a CS major and I am a post baccalaureate student. I have experience with Java, C++, C# and HTML. The area that I am interested in is game design. I also have experience in making a game in unity as that was the focus of a group I had worked with in the past. My main focuses in that project were projectile mechanics and some inventory mechanics. I have worked with groups in the past over multiple subjects so there’s a lot I can bring to this team.

We have spent 5 hours a week each on the requirements. This time has been spent both on discord and in Raymond Cravens library planning and working on our draft and presentation. The days we have been working jointly are on Tuesdays and Thursdays.

# Technical feasibility

The technicality of our project includes unity version 2020.3.17f1 and C# version 8.0. It should be feasible to implement C# scripts since all group members are familiar with scripting and programming. We are also vaguely familiar with using Unity so learning and understanding both Unity’s development environment and the C# programming language shouldn’t be a risk in our project to remake the NES game Contra. An added guarantee of the project's feasibility using Unity is that we will all complete Unity tutorials and read documentation before writing source code. The database portion will utilize php my admin and my sql for the saved game data.

# Schedule feasibility

To get everything done in an orderly manner, and to see how much we are getting done our team will be using two tools. First is the gantt chart and another is the activity graph. Gantt chart will cover the whole project and will give the client and other stakeholders a better view of how the team is preparing and performing the tasks and what they are planning next. An activity graph will be how the team is adding those pieces together and how much time they are spending in doing that as a team. Let's first talk about the project and our estimated time to complete the project.

Duration

Our project is divided into four sprints. Each sprint will be about 3 to four weeks long. We will be following the …….. Process model to complete the project. It means we will complete task1 and then start on the next task.

* Our initiation of the project was on August 23rd when sprint one started and will end on September 12th. During this sprint we will be gathering all the requirements from the client and other sources like previous creation and similar works. Then our other task during this sprint will be team organization and feasibility study. We will familiarise ourselves with the Unity Game engine interface on which we will be creating the game. And also during all this we will be keeping and updating the documentation so it will be easier for all stakeholders involved in the project to know what's going on and later after the project is finished for maintenance purposes.
* Second sprint will begin on September 13th and will end on October 10th. During the sprint 2 we will design the different aspects of the game and again keep the complete documentation of what we are doing.
* Third sprint will begin on October 11th and will end on November 7th. During the sprint 3 we will be implementing the actual game using the unity game engine developer and again keep the complete documentation of what we are coding.
* Final sprint will last for the remainder of the semester and will consist of testing of our product, and changing what needs to be changed in the game and keeping all the record of how we achieved desired results, and what mistakes we made along the way.

Current Sprint

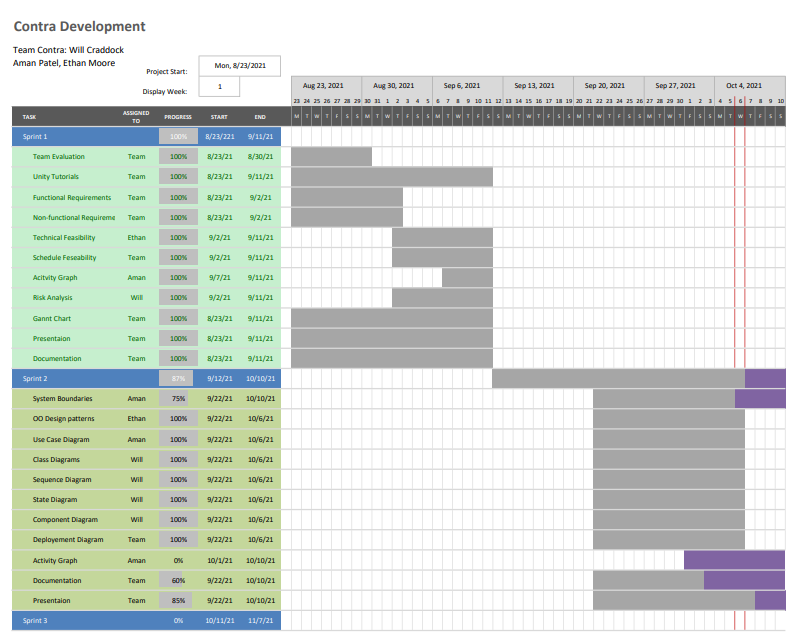
Estimated time for each task:

* Object oriented diagrams took about two and a half weeks to complete.
* Behavioral UML diagrams took around two and a half weeks to complete.
* Structural UML diagrams took around two and a half weeks to complete.
* Wire frame diagrams took around 6 hours to complete.
* Activity graph and Gantt chart took 10 hours altogether to complete.

During this sprint we will mostly work on different diagrams. Basically design how we will structure our game.

# Progress visibility

Gantt Chart:



Activity Graph:



# Risk analysis.

Risk analysis is something that most businesses take seriously. This has not changed in our group project. We have looked at this from different angles to ensure that we have a plan if something were to go wrong. We have looked into potential risks that could come up in the future and have provided some examples.

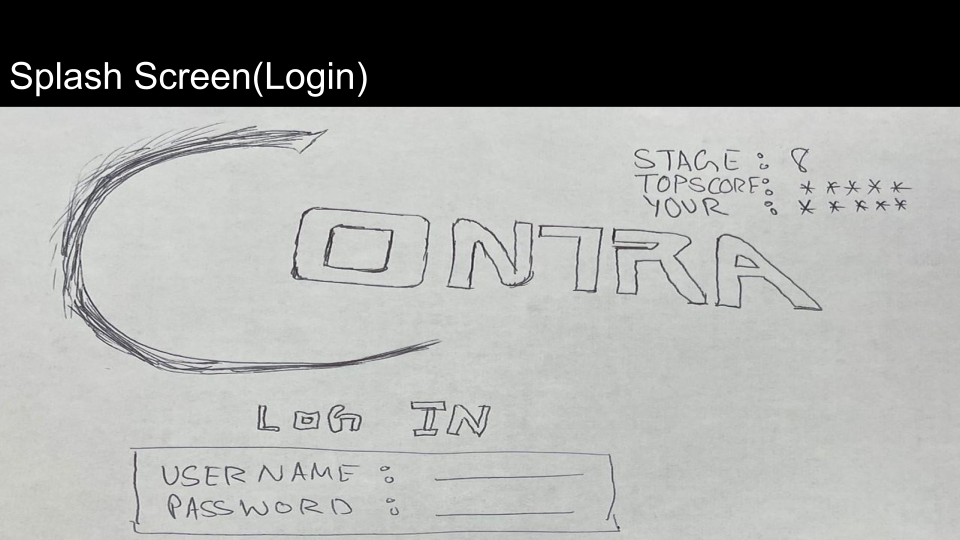
Potential risks

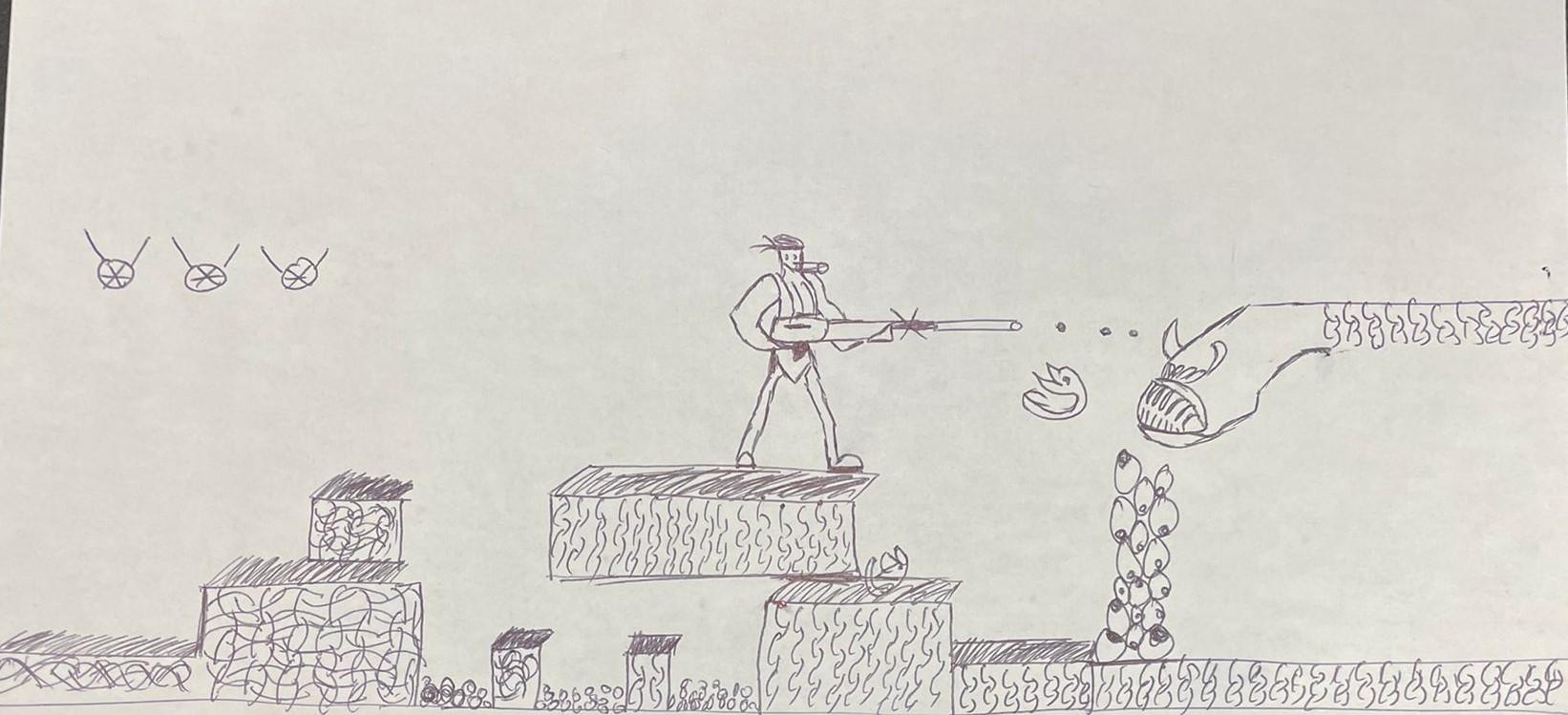
* Hardware malfunctions
* Incompatible code/code malfunctions
* Schedule availability
* Physical health
* Extreme weather
* Unknown factors

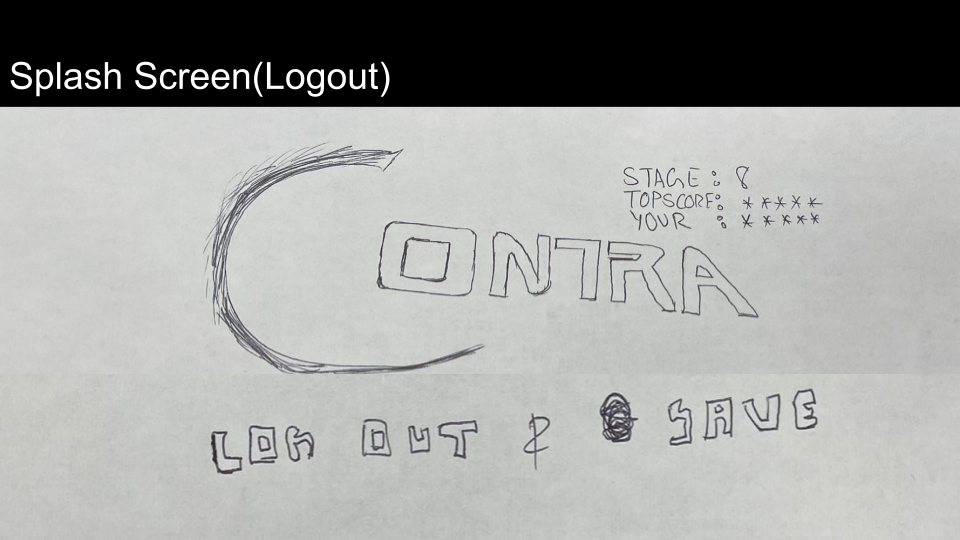
Looking at these risk factors the risk with the most potential to occur is most likely a piece of hardware or software being incompatible or malfunctioning. Rest assured we have a contingency plan if any of these risks were to come up.

## 

# Wireframe diagrams/Storyboards



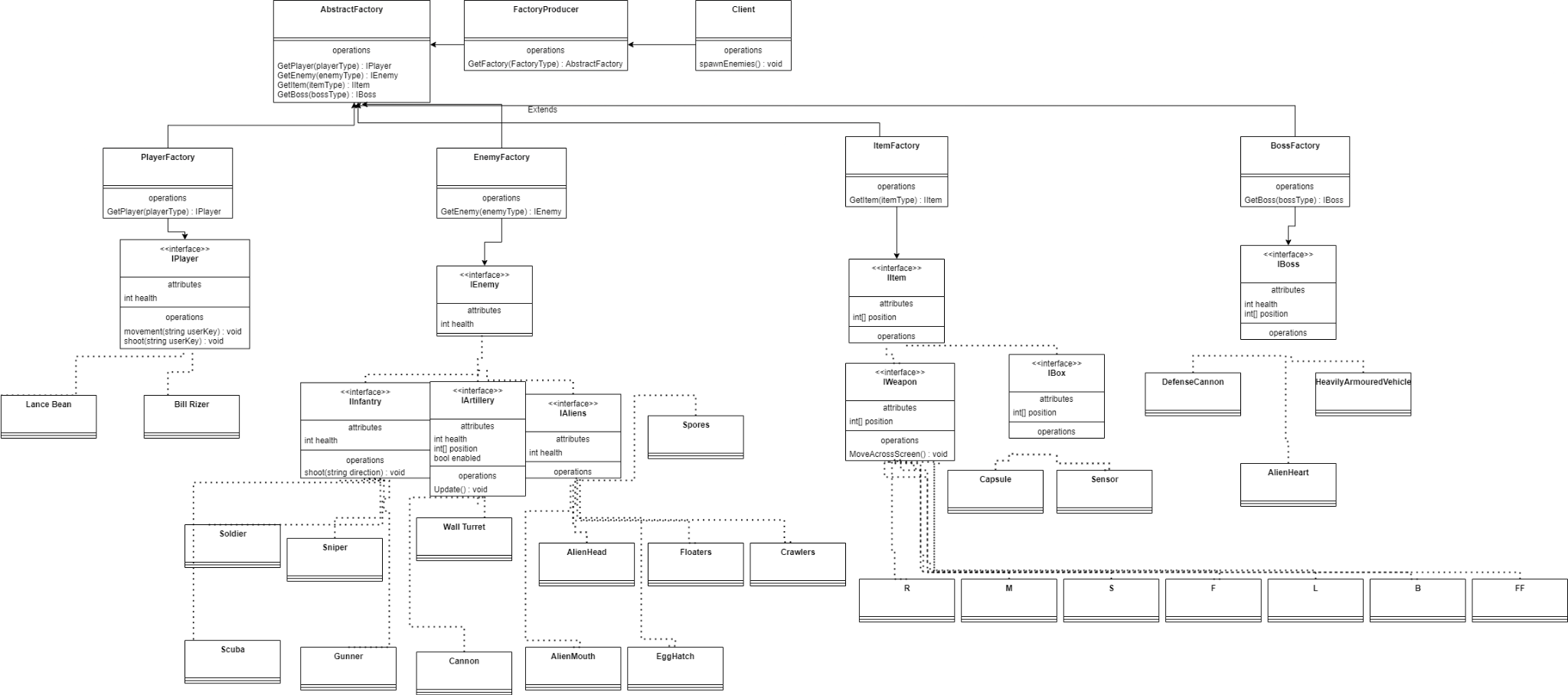




Our wireframe diagram above is showing how the user interacts with the start screen and logging into the database. The user then plays the game and in the picture above is attacking the final boss of Contra. After defeating the final boss or if the player gives up the user logs out and saves. The current users information is shown next to the Contra logo stating the stage number, score, and username.

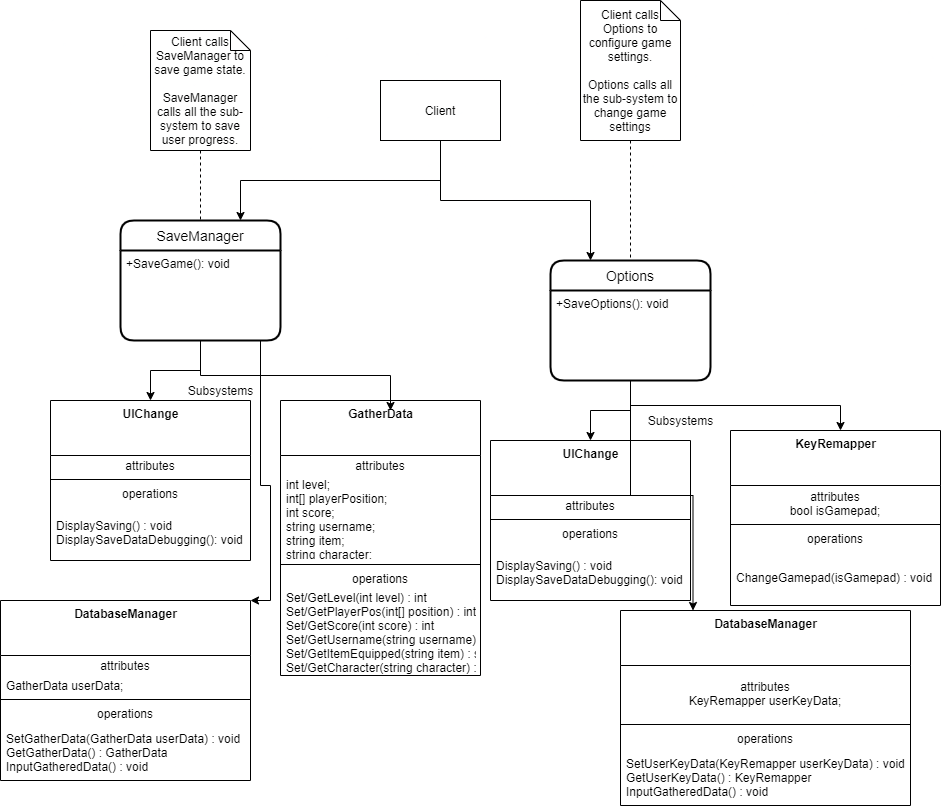
# OO Design Patterns

## Creational - Abstract Factory



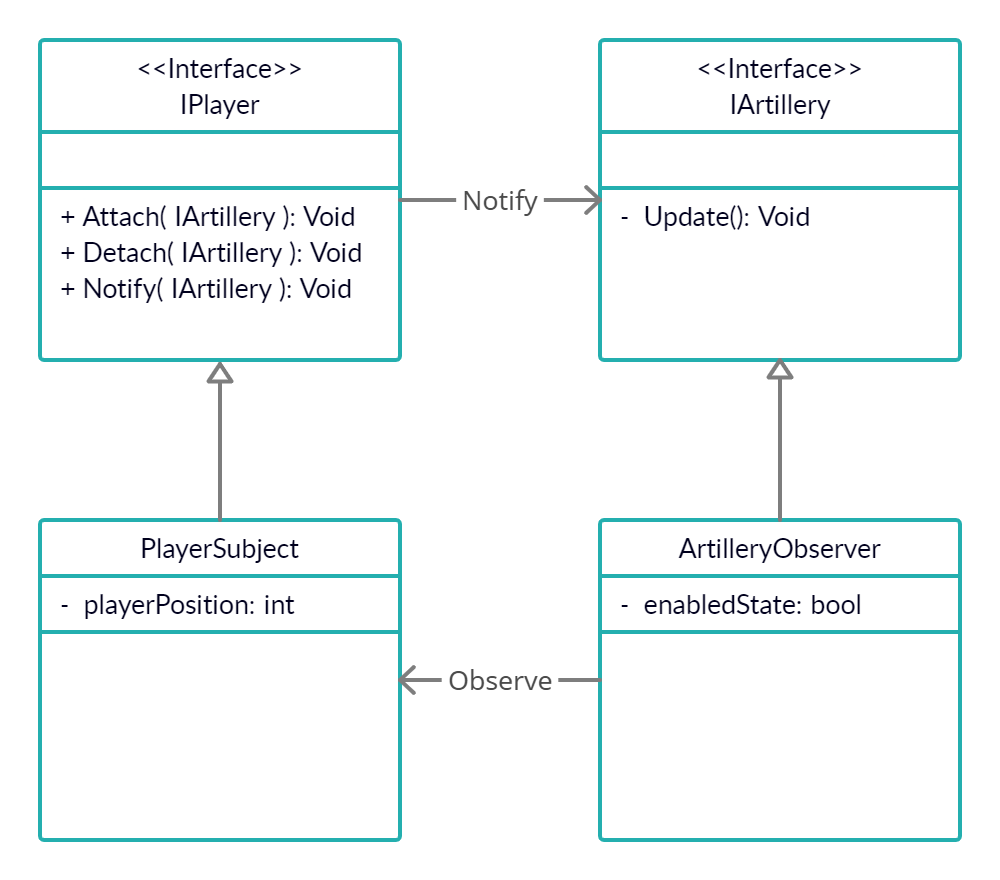
The diagram above is the creational object oriented design pattern. The abstract factory design pattern was chosen because the different types of game objects in the game need to have their own groups but also be within a general factory. If the centralizing of our gameobjects creates too much code complexity then we can separate the separate game objects into their own factories. The different object factories are PlayerFactory, EnemyFactory, ItemFactory, and BossFactory.

## Structural - Facade



The diagram above is the facade structural object oriented design pattern. The facade design pattern was chosen because the save manager and options subsystems are sufficiently complicated and can benefit from having an accessible entry class. The save manager is using the SaveManager class to access the UIChange, GatherData, and DatabaseManager classes. The subsystem will gather data, input into the database, and notify the user via an icon appearing noting that their game state has been saved. The options menu is using the Options class to access the UIChange, KeyRemapper, and DatabaseManager classes. The subsystem allows the user to make changes to the keys or other settings, input the setting data into the database, and notify the user using the UIChange class.

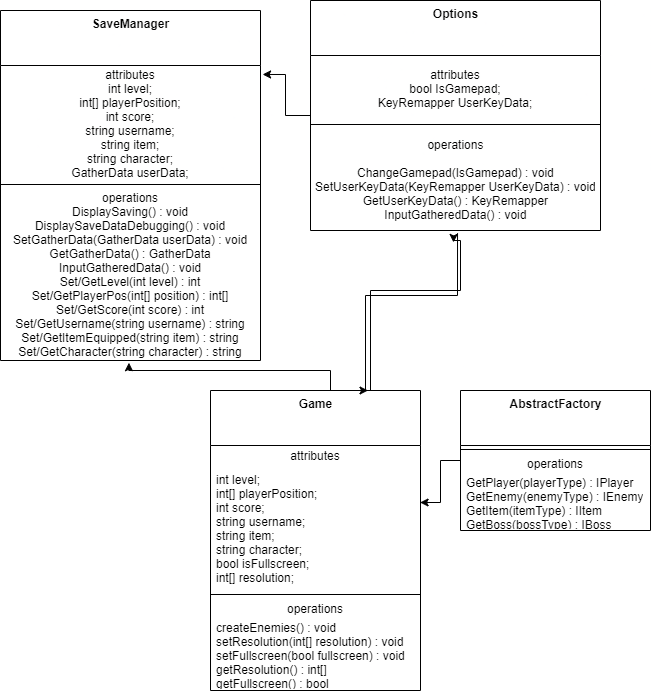
## Behavioral - Observer



This diagram is the observer behavior object oriented design pattern. The observer design pattern was chosen because the Artillery game objects are activated only when the player is within a certain distance from them. The observer pattern watches for the player's x-axis position relative to them and a certain threshold activates the artillery gameobjects methods. The player knows that the observer pattern has activated the enemy when the sprite model changes and the enabledState attribute is set to true.

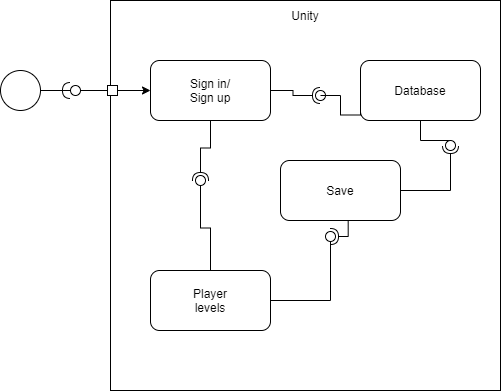
# Structural UML Diagram

## Class Diagram



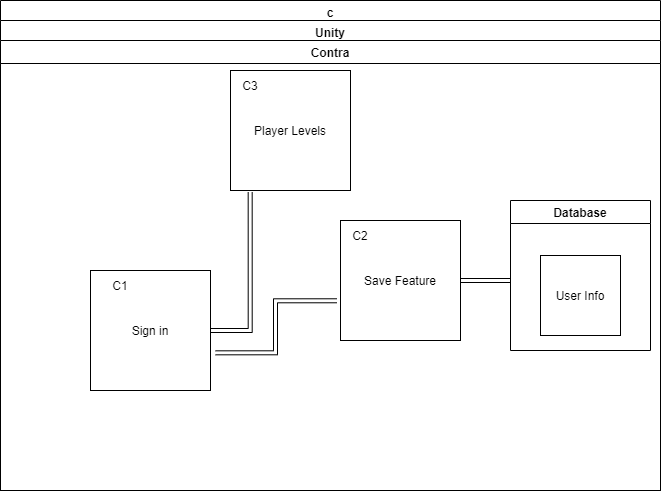
The diagram above is our class diagram for our remake of Contra. The game classes have been abstracted into the classes above. The Game class has attributes like the score and username of the player which gets gameobjects like the player and enemies through the abstract factory design pattern. The Game class calls on the Options and SaveManager facades to save game states to the database.

## Component diagram



The Contra component diagram outlines the interactions between the components like the sign in function, the database, and player levels. The user interacts with the system through the sign in/up component.

## Deployment diagram



The deployment diagram is outlining the interactions between the components of the game levels and saving to the database. These components are interacting within the unity engine and the user's computer allocating the necessary resources.