**Software Engineering**

**Analysis Document**

**Birds of Prey**

Team 6 Members:

Sebastian Farias - Team Leader and Technical Writer

Maaz Ansari - Configuration Manager

Anibal Ruiz - API - Liaison

Jean Claude Charles Boute - Webmaster

**Table of Contents**

1. Introduction

1.1 Objective

1.2 What is Birds of Prey?

1.3 User Playability

1. Overview
2. Proposed System

3.1 Functional Requirements

3.1.1 Play Game

3.1.2 Single Player

3.1.3 How to Play

3.1.4 Credits

3.1.5 Close Game

3.2 Extendibility

3.2.1 Pause the Game

3.2.2 Display Instructions

3.2.3 Artificial Intelligence

3.3 Nonfunctional Requirements

3.3.1 Game Performance

3.3.2 Usability

3.3.3 Supportability (Maintainability)

3.3.4 Security

3.3.5 Reliability

3.3.6 Response time

3.4 Constraints (Pseudo Requirements)

1. System Models & UML Diagrams

4.1 Use Case Diagram

4.1.1 Use Case 1: Start Game

4.1.2 Use Case 2: Pause Game

4.1.3 Use Case 3: How to Play

4.1.4 Use Case 4: Display Instructions

4.1.5 Use Case 5: Credits

4.1.6 Use Case 6: Display Credits

4.1.7 Use Case 7: Back to Menu

4.1.8 Use Case 8: Close Game

4.2 Sequence Diagrams

4.2.1 Scenario A) Player starts game and launches bird

4.2.2 Scenario B) Player completes level

4.2.3 Scenario C) Player fails level

4.2.4 Scenario D) Player selects How to Play

4.2.5 Scenario E) Player selects Credits

4.3 Activity Diagram

4.3.1 Activity Diagram Explanation

4.4 Class Diagram

4.4.1 Class Diagram Explanation

1. User Interface

5.1 Main Menu

5.2 Game Screen

5.3 Pause Menu

5.4 How to Play Screen

5.5 Credits Screen

5.6 Art

1. Glossary & References
2. Team Member Contributions
3. **Introduction**

**1.1 Objective**

Our team’s objective is to recreate the mobile game “Angry Birds” that is playable on IOS and Android devices, and make it playable on desktop and laptop devices. We will also incorporate a different art style for the birds, enemies, tower structures, environments, and also create our own level layouts. Our version of the game will be called “Birds of Prey”.

**1.2 What is Birds of Prey?**

Birds of Prey is our team’s vision of “Angry Birds”, the mobile game available on IOS and Android devices, but our version will be playable on desktop. In Birds of Prey, there are thieves and other enemies that have invaded the forest in which the playable birds live; the thieves are trying to claim the forest as their territory but the birds will not back down. In order for the birds to drive the thieves out of their home, they must launch themselves towards their enemies and destroy them and their tower structures.

**1.3 User Playability**

This game is a single player game and the user will be able to control the birds and launch them over to destroy their enemies. The user will have to estimate the angle and the distance at which to launch the birds in order to successfully complete the mission; the user will also have a limited number of birds in order to do so. There will be various birds in which the user can choose to use as well as various enemies that the user will have to face. The user can simply play the game solely with a mouse or a trackpad if the user is playing on a laptop.

1. **Overview**

Section 3 of the Analysis Document will include the proposed system which consists of functional requirements, nonfunctional requirements, and constraints (pseudo requirements). Then section 4 of the document will explain the system models of the software system; this will include scenarios, use case diagrams, sequence diagrams, class diagrams, and activity diagrams. Section 5 will analyze the user interface of the system and section 6 will be the glossary and references of the document. Finally, section 7 will include a report of the responsibilities of each team member and their contribution to this document.

1. **Proposed System**

**3.1 Functional Requirements**

A functional requirement defines a system or its components. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Also functional requirements are able to specify particular parts of a system. Part of a functional requirement is the objective of the game, which is a very important part of the game, objective helps you with verifying the functionality of the game.

**3.1.1 Play Game**

For now, there will be one mode game: Single player.

**3.1.2 Single player**

The user enters at this screen in the main menu of the game. By entering in the single player, the user may be able to start the game. The player will play alone offline.

**3.1.3 How to Play**

This screen can be accessed from the main menu. The system will display the backstory of the game, and will also display the instructions on how to play the game as well as the controls of the game. The controls include mouse dragging and trajectory, and keyboard controls.

**3.1.4 Credits**

The names of developers and programmers may appear here. 'End game screen' stats from the game or development, so as well as showing who worked on the game you show stats like how many coins the player collected or how many levels the player completed throughout the game.

**3.1.5 Close Game**

The Close Game option on the main menu will terminate the game and return the player to their desktop.

**3.2 Extendibility**

The actions for the player are simple; the player can operate the program by using a mouse/trackpad, and a keyboard.

**3.2.1 Pause the game**

The player will be able to pause the game. The pause menu will include options such as resume game, return to menu, and how to play the game.

**3.2.2 Display Instructions**

The player will be able to display the instructions of the game if he/she forgets the rules. The last thing you need to put in is a set of instructions to help others learn how to play. Teaching a completely new game to the public is not always easy. It is important to remember that your audience has no idea how any aspect of your game works yet. That's where your set of game rules comes in. Writing game rules can take a bit of time. But it's important to include detailed instructions of the objective, all the pieces, and how the game is played.

**3.2.3 Artificial Intelligence**

We are planning to make the enemies move randomly. In Birds of Prey, there are thieves and other enemies that have invaded the forest in which the playable birds live; the thieves are trying to claim the forest as their territory.

**3.3 Nonfunctional Requirements**

Defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. For example, how fast does the system load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Nonfunctional requirements are just as critical as the functional requirements, they are set to make sure that the games usability and effectivess.

**3.3.1 Game performance**

Birds of Prey is a single-player game. This is an advantage because the player will not have to wait for other players to play or to compare the scores. We still need to design, program and test the project for a better understanding of the performance of the game in specific conditions. Our plan is to make the game requirements are not too high but not too low due to the fact that we need to make sure that the game is able to be compatible with most computers as well as be able to meet the deadlines of deliverables. We are trying to make sure that the game that we want to create does not have too many bugs or giving to many issues, we want to make sure that it is neat and understandable.

**3.3.2 Usability**

The simplicity of the user interface will be essential. Users with no technical background or knowledge may be able to enjoy the game. There are many reasons why usability is important in games. For one, playing games is voluntary. If the player has to struggle with problems that make playing less fun than doing something else, then there is nothing to stop the player from switching off the game. This is a serious risk as the user experience is very sensitive to usability problems. Even the smallest glitch or hiccup in the user interface may render an otherwise good game into a rather annoying experience.

**3.3.3 Supportability (Maintainability):**

Birds of Prey will be coded in a maintainable way, in other words, we will make it simple for other programmers to understand the code and the implementation. Usually, while playing your game, a lot of players come up with ideas and suggestions which you can use in the future. We are sure that there are no inviolable rules of game design and that the only guiding line for the accuracy of our decisions is the expectations of our target audience, which is why user feedback is so necessary. Once we are able to get more feedback and interaction towards the game it helps us to have a better understanding of what the target audience wants in a game.

**3.3.4 Security**

This game will be available to play on desktop and laptop devices and can be downloaded by anyone no matter where they are. From the cyber perspective anyone that installs this app their information is secured from hackers or anyone else that is trying to retrieve their information from the game. Our goal is to make a fun, safe game that everyone can play no matter the age since it is a general game, where some other games would need parental guidance.

**3.3.5 Reliability**

The game progress of a player will be saved locally so that the user can load his/her game and continue from where he/she left. Records will be saved in an online database. This game progresses and records will not be lost because of a system failure or a power loss. For a lot of gamers, a poor connection or unreliable services can be one of the biggest deal-breakers. A game that doesn’t run properly or that won’t even start can quickly lose a fanbase.

**3.3.6 Response time**

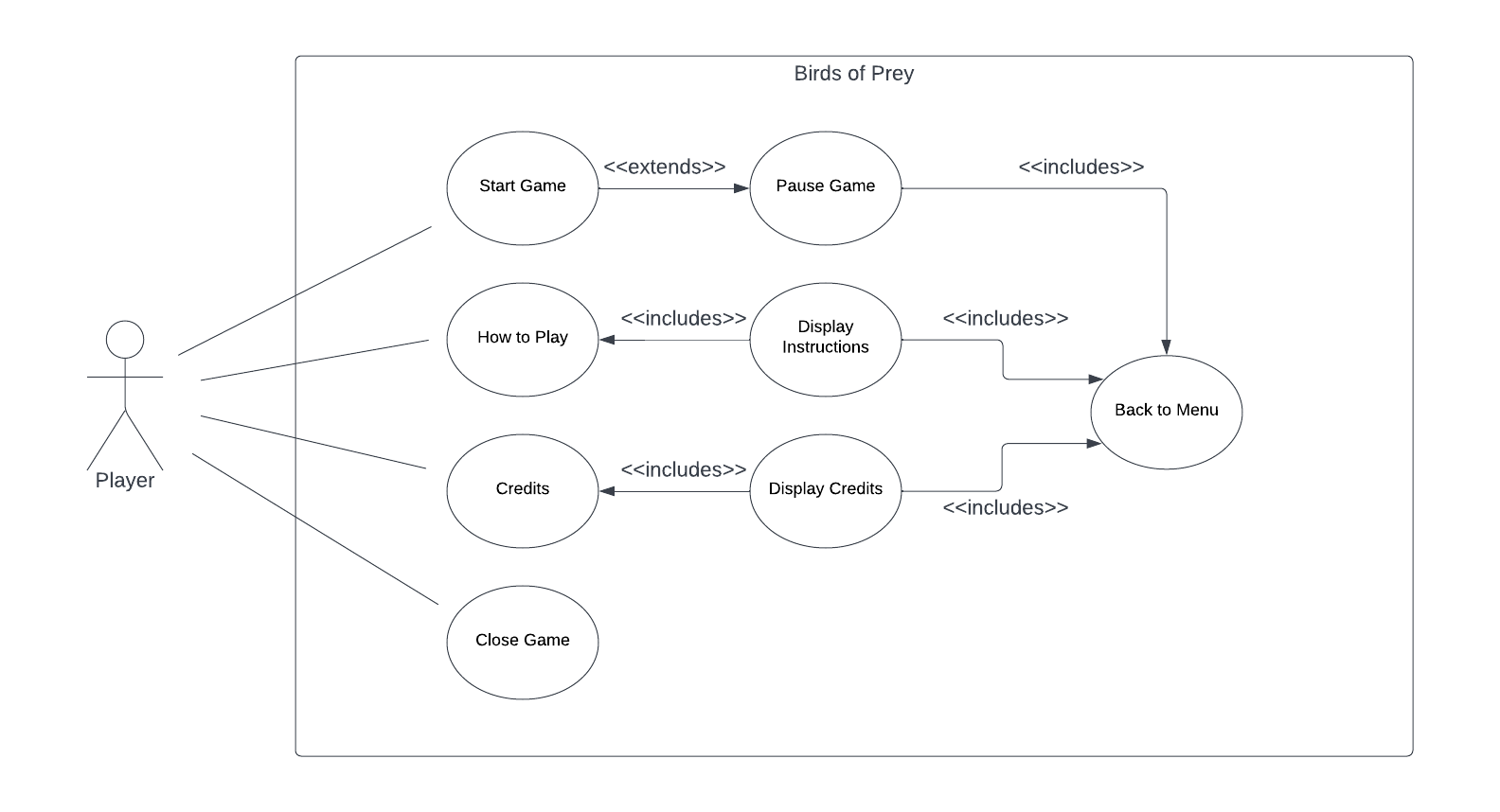
The sharper image that can show moving objects with little-to-no blur makes it easier to see and flick to your targets accurately. Lower response time is great for helping reduce some motion blur on the screen that is caused by higher response times.Anything in the range 1 - 5 milliseconds is a great time for gaming. We’ll try to optimize our game, so we can have the lowest response time. The lower the response time the better the outcome of the game. Lower response time is good because it helps with reducing some motion blur on the screen which happens when it has higher response time, with visuals being between 1ms and 5ms you won't really be able to notice the difference. To have a great working game you never want to go pass 5ms we will try our best to stay between the range listed above.

**3.4 Constraints (Pseudo Requirements)**

Birds of Prey will be coded in C# and will use the Unity engine in order to run the game. We will also use video tutorials on youtube in order to help us to create the game and the necessary features (Our team has received professor approval to do so).

1. **System Models & UML Diagrams**

**4.1 Use Case Diagram**

****

**4.1.1 Use Case 1: Start Game**

Use Case Name: Start Game

Participating Actor: Player

Entry Conditions:

Player Chooses the “Start Game” option on the main menu.

Exit Conditions:

Player completes all levels of the game or selects “Back to Menu” from the pause menu.

Flow of Events:

1. Player selects “Start Game” from the main menu.
2. System loads the first level of the game.
3. The player can use their mouse or laptop trackpad to launch a bird from the starting point.
4. The game camera follows the bird as the player launches it.
5. The enemies and tower structures are knocked down and destroyed.
6. The player continues to launch birds until there are no more enemies left.
7. Once all enemies are destroyed the system advances to the next level.

Special Requirements: None.

**4.1.2 Use Case 2: Pause Game**

Use Case Name: Pause Game

Participating Actor: Player

Entry Conditions: Player clicks on the “Pause Game” button on the screen.

Exit Conditions: Player clicks on the “Pause Game” button again to return back to the game.

Flow of Events:

1. Player selects the “Pause Game” button from the screen.
2. The system pauses the game.
3. The player cannot move the characters or play the game during this time.
4. Player selects the “Pause Game” button again in order to continue playing the game.

Special Requirements: None

**4.1.3 Use Case 3: How to Play**

Use Case Name: How to Play

Participating Actor: Player

Entry Conditions: Player clicks on the “How to Play” button on the main menu.

Exit Conditions: Player clicks on the “X” button on the top right of the dialogue box.

Flow of Events:

1. Player initiates the game and is brought to the main menu.
2. Player clicks on the “How to Play” button from the following menu.
3. System loads up a textbox with instructions on how to play the game and will include a brief backstory of the game.

Special Requirements: None

**4.1.4 Use Case 4: Display Instructions**

Use Case Name: Display Instructions

Participating Actor: Player

Entry Conditions: Player chooses the “Start Game” option on the main menu and proceeds to click on the button on the bottom left titled “Instructions”

Exit Conditions: Player clicks on the “X” button on the top right of the “Instructions” text box.

Flow of Events:

1. Player chooses the “Start Game” option on the main menu and proceeds to click on the button.
2. Player chooses the “Instructions” option on the following menu that pops up.
3. A text box will appear with instructions for the current level.
4. The player can click on the “X” button on the top right of the “Instructions” text box.

Special Requirements: None

**4.1.5 Use Case 5: Credits**

Use Case Name: Credits

Participating Actor: Player

Entry Conditions: Player chooses the “Credits” button on the main menu.

Exit Conditions: Player clicks on the “X” button on the top right of the “Credits’ button.

Flow of Events:

1. Player chooses the “Credits” button on the main menu.
2. System loads up a text box with the credits.
3. The player can exit by clicking on the “X” button on the top right of the text box.

Special Requirements: None

**4.1.6 Use Case 6: Display Credits**

Participating Actor: Player

Entry Conditions: Player chooses the “Credits” button on the main menu.

Exit Conditions: Player clicks on the “X” button on the top right of the “Credits’ button.

Flow of Events:

1. Player chooses the “Credits” button on the main menu.
2. System loads up a text box with the credits.
3. The player can exit by clicking on the “X” button on the top right of the text box.

Special Requirements: None

**4.1.7 Use Case 7: Back to Menu**

Use Case Name: Back to Menu

Participating Actor: Player

Entry Conditions: Player clicks on the “Back” button on the bottom left of the dialogue boxes.

Exit Conditions: Player clicks on the “X” button on the top left of the dialogue box to return back to the main screen of the game.

Flow of Events:

1. Player clicks on the “Back” button on the dialogue box.
2. Game goes back to the main menu.

Special Requirements: None

**4.1.8 Use Case 8: Close Game**

Use Case Name: Close Game

Participating Actor: Player

Entry Conditions: Player clicks on the “Exit” button on the main screen.

Exit Conditions: Player clicks on the “Start Game” option in order to not exit.

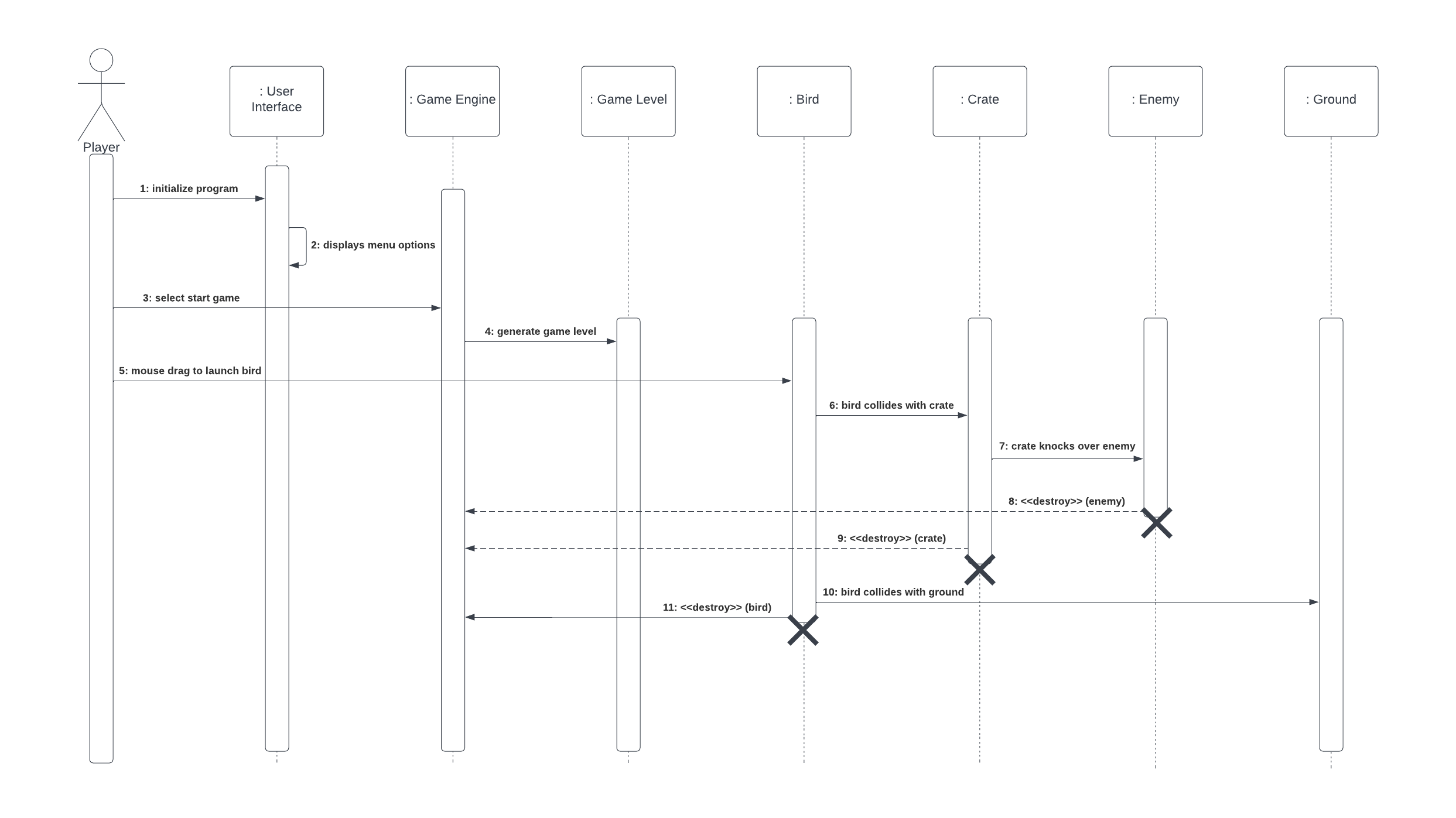
Flow of Events:

1. Player clicks on the “Exit” button on the main screen located on the bottom of the screen under the play game button.
2. Game exits and is closed; the player will then return to their desktop.

Special Requirements: None

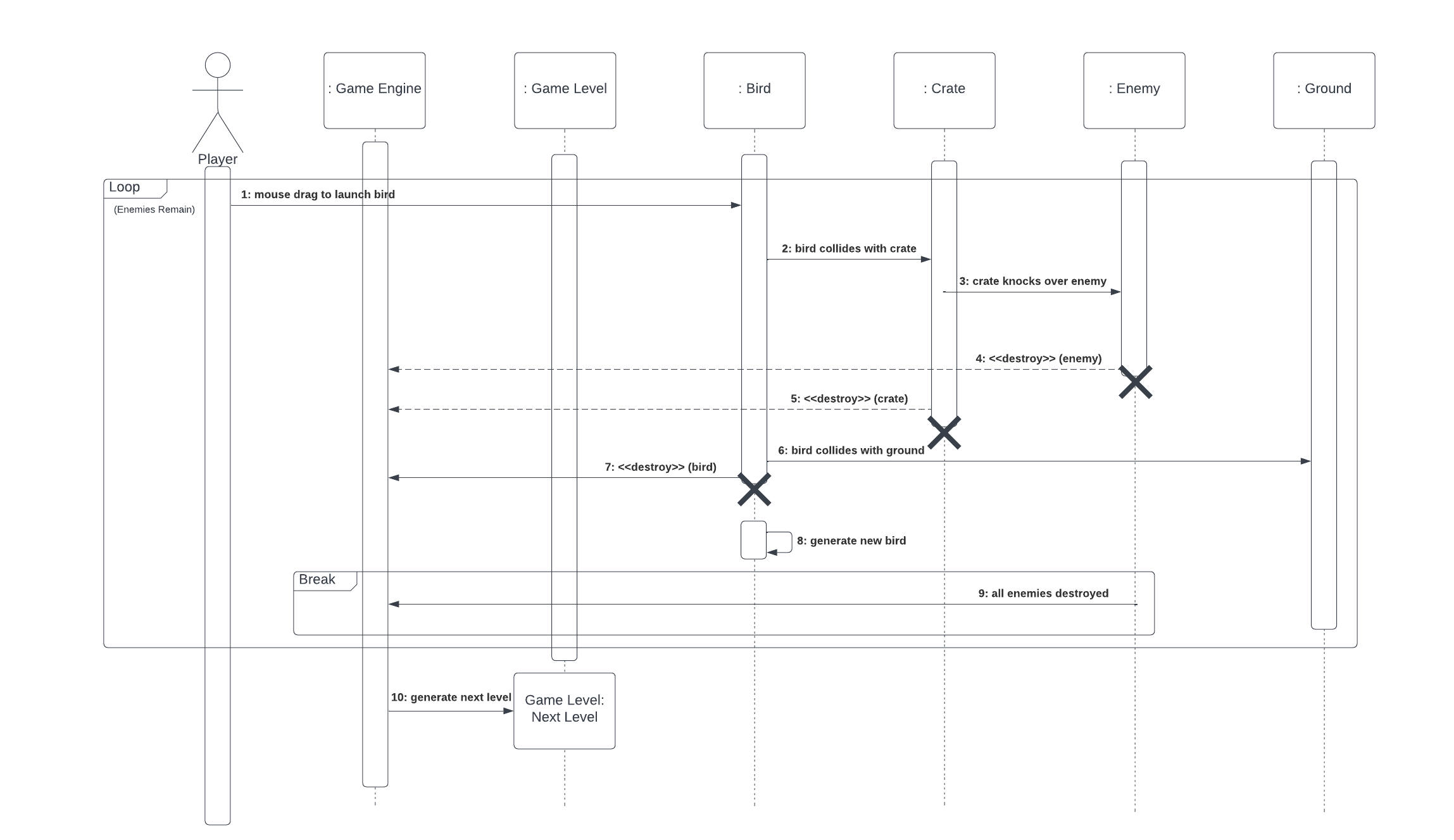
**4.2 Sequence Diagrams**

**4.2.1 Scenario A) Player starts new game and launches bird**

****

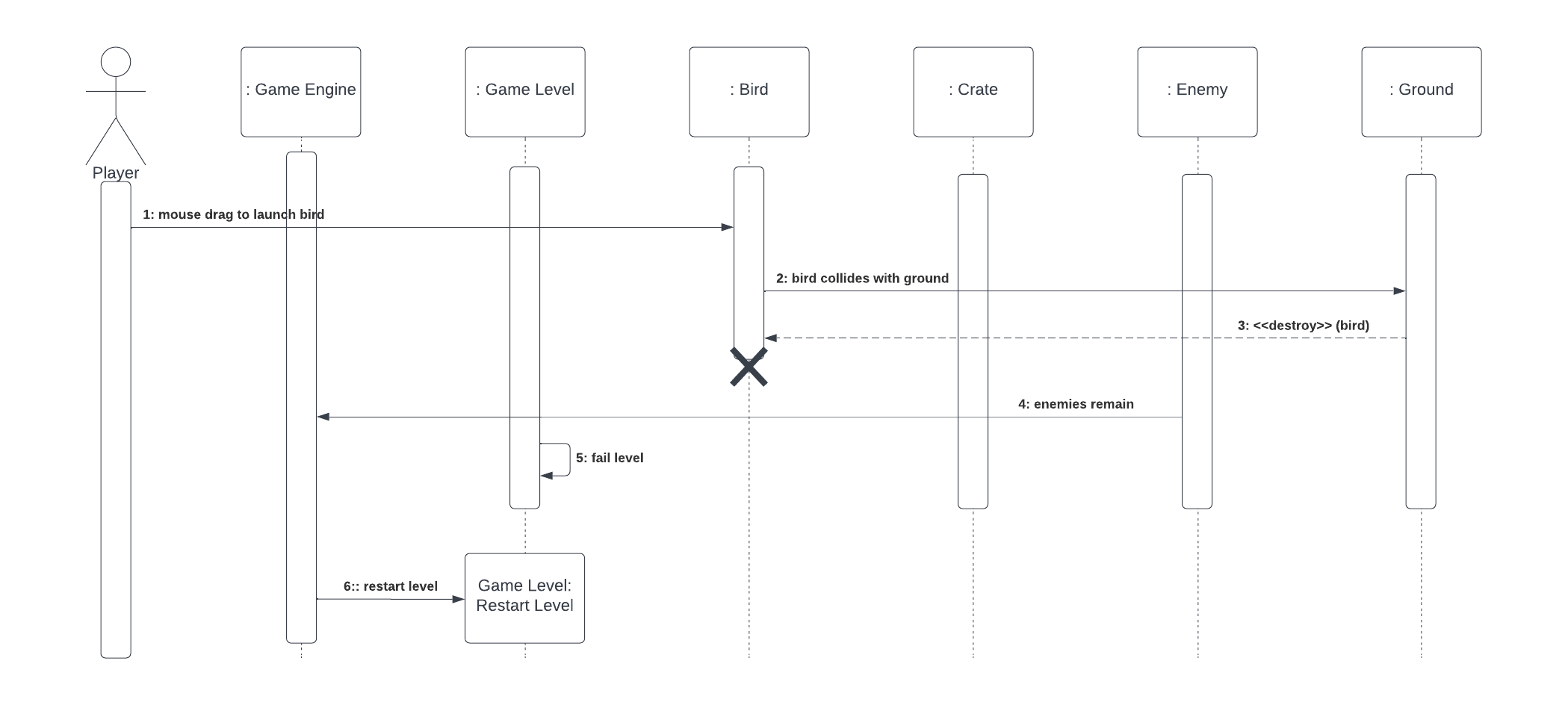
Explanation: The player wants to start the game. The game engine displays the menu options. He/she enters the Main Menu and chooses one of the modes: single game. The game engine starts the game level, then the player drags the bird with the mouse and starts playing the game. After that, the bird is launched and collides with the crate, and this crate knocks over the enemies. The enemies are killed or eliminated, the crate is destroyed. Then if the bird collides with the ground, the bird is destroyed.

**4.2.2 Scenario B) Player completes level**

****

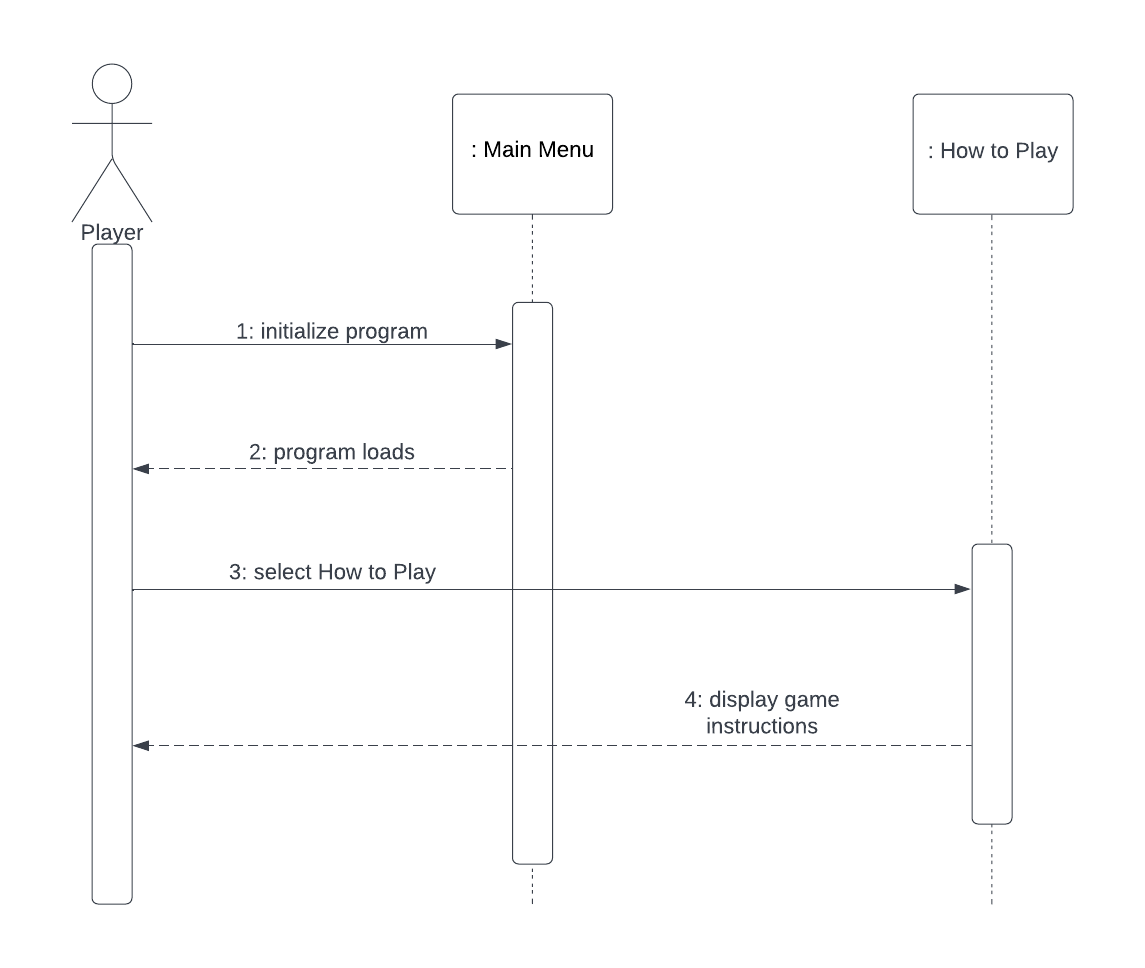
Explanation: The game engine starts the game level, then the player drags the bird with the mouse and starts playing the game. After that, the bird collides with the crate, and this crate knocks over the enemies. The enemies are killed or eliminated, the crate is destroyed. Then if the bird collides with the ground, the bird is destroyed. The engine game generates a new bird to continue with the game. When all enemies are eliminated, it breaks and the level is passed. The next level is then generated and ready to play.

**4.2.3 Scenario C) Player fails level**

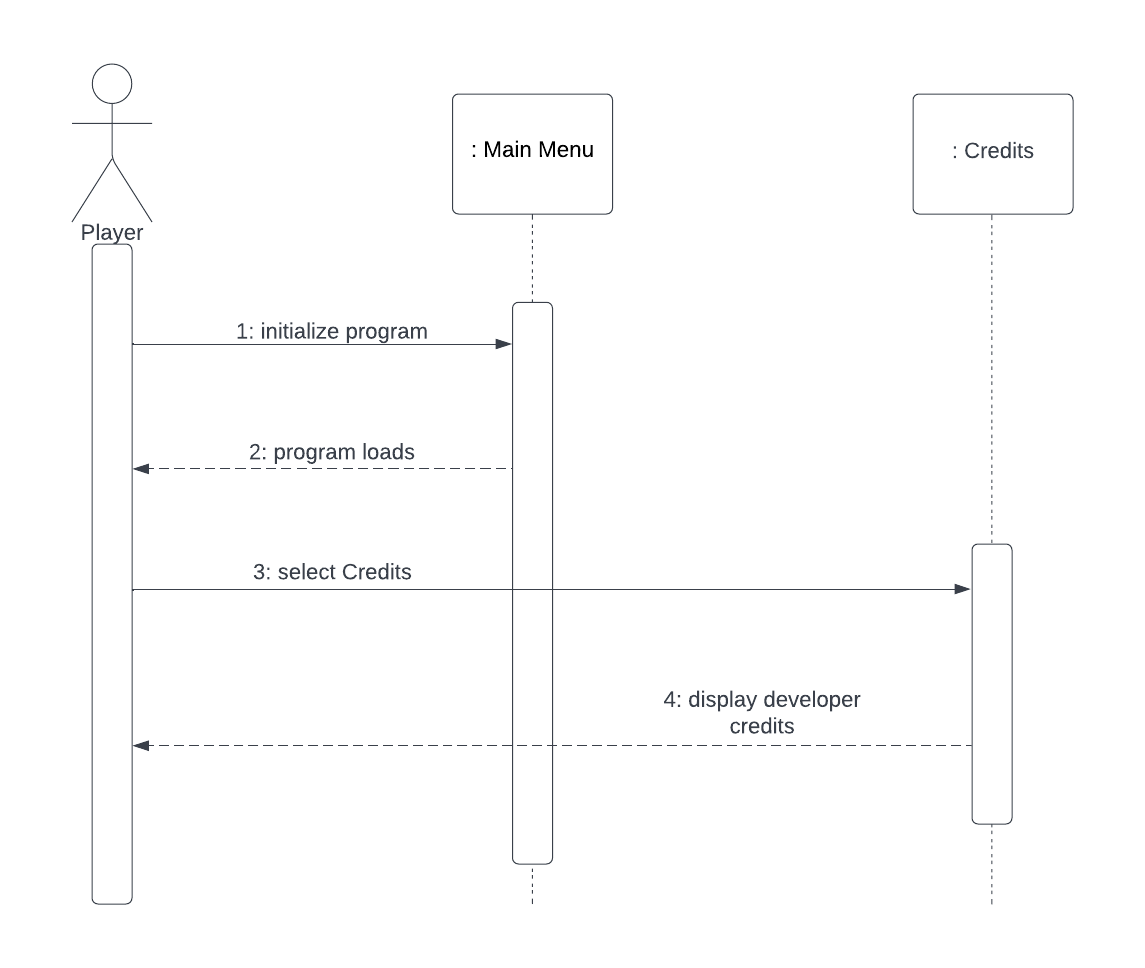
****

Explanation: The game engine starts the game level, then the player drags the bird with the mouse and starts playing the game. After that, the bird collides with the crate, and this crate knocks over the enemies. The enemies are killed or eliminated, the crate is destroyed. Then if the bird collides with the ground, the bird is destroyed. The engine game generates a new bird to continue with the game. If no more birds are left and all enemies are still alive, the player fails the level. The player then has the option to restart the level and continue to play.

**4.2.4 Scenario D) Player selects How to Play**

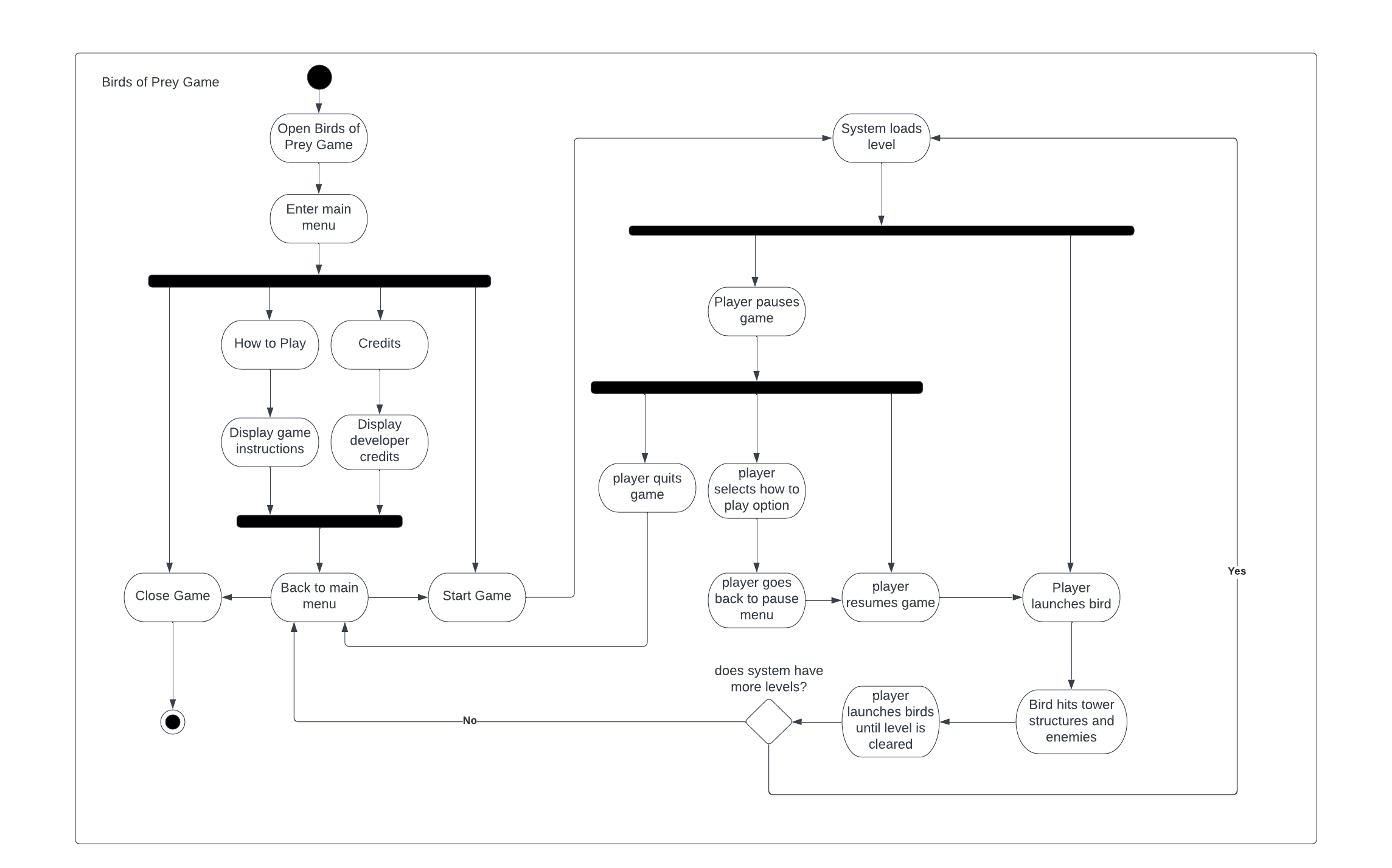
****

Explanation: The player initializes the program and the main menu is displayed. After the program loads, the player selects the option How to Play and the system displays the game instructions.

**4.2.5 Scenario E) Player selects Credits**

Explanation: The player initializes the program and the main menu is displayed. After the program loads, the player selects the option Credits, and the system displays the developer credits with all the information.

**4.3 Activity Diagram**

****

**4.3.1 Activity Diagram Explanation**

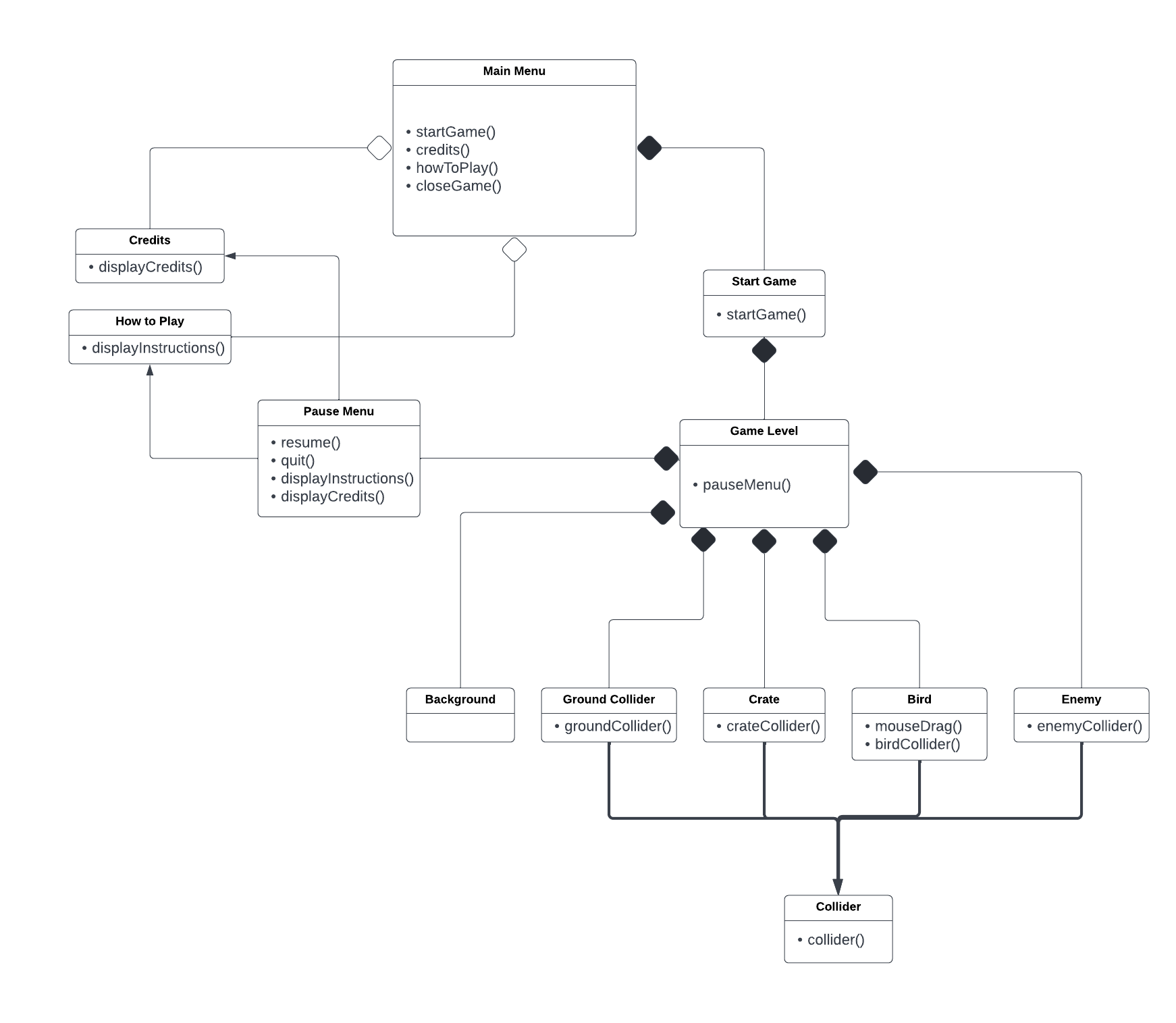
The activity diagram above shows how the player or the user flows through the system from running the game all the way until they choose to exit the game. First, the player runs the game and starts at the main menu where it gives the player the option to either start the game, learn how to play, display the credits, and close the game.

The player can press the “How to Play” button before actually playing the game; after the player reads the instructions and a brief introduction to the story, the player will have the option to redirect themselves back to the main menu. The player can also press the “Credits” button to view the developers of the game, and similarly to the how to play option, they can redirect themselves to the main menu by the button that says “Back to Main Menu”.Once the player is back to the main menu, they can either press the “Close Game” button which will completely close the application, or start the game. Once the player presses the “Start Game” button the system will load the first level of the game.

Once the player loads into the first level, they have the option to play the game and launch the first bird, or pause the game. The player can pause the game at anytime; if the player pauses the game, they can return to the main menu with the “Quit Game” button, press the “How to Play” button to view the instructions, or press the “Resume Game” button which will resume the current level and at the time it was initially paused.

The player starts playing by launching a bird towards the enemies and their tower structures. If the player uses all birds and eliminates all enemies of the level, the system will then check if there are more levels or if the player has completed all levels. If the player still has more levels to complete, they will be loaded into the next level. If the player has completed the game, they will be redirected to the main menu and be shown the title screen with the four buttons. In certain parts of the game even after they’ve finished playing the game they will receive certain rewards before returning to the main menu.

**4.4 Class Diagram**

****

**4.4.1 Class Diagram Explanation**

This is a bare-bones class diagram of the program to give our team an idea of what classes and objects we need to incorporate to our program.

* Main Menu: will include start game, how to play, credits, and close game
* How to Play: will display controls and backstory of the game
* Credits: will display developer credits and references used
* Close Game: will terminate the program
* Start Game: will initialize the game and load the game level
* Game level will load the following objects:
* Background, Ground, Crate, Bird, and Enemy
* These objects will inherit a collider class in order for the game engine to detect the boundaries of the game objects (apart from background)
* The pause menu will allow the player to resume, quit, and view the credits and how to play
* The pause menu will inherit the How to Play and Credits options from the main menu

1. **User Interface**

**5.1 Main Menu**

The main menu will have art from the game and will also display the title of our game. This screen will also display options including “Start Game”, “How to Play”, “Credits”, and “Close Game”. The player will be able to access these options by hovering and clicking them with the mouse or trackpad. The “Start Game” button will load the player into a single player game. The “How to Play” button will display instructions on how to play the game, and the “Credits” button will display the developers of the software and references used.

**5.2 Game Screen**

The game screen will display a mountain range as a background, a grass ground, and imported art for the birds, enemies, and tower structure.

**5.3 Pause Menu**

The pause menu will only be accessed during the game. This feature will allow the player to pause the game for external reasons; the player then has the option to resume the game, quit the game, and read the instructions on how to play the game.

**5.4 How to Play Screen**

The how to play screen will display the story of the game and the instructions on how to play the game to the player. After the player is done reading the instructions, the player then will press “Back to Menu” in order to access the Main Menu

**5.5 Credits Screen**

The credits screen will display the credits of the game; this includes the names of the developers, their roles, and other references used in order to create the game (i.e. websites of imported art.).

**5.6 Art**

The game art including the environment, birds, characters, and tower structure will be imported from the following websites:

<https://assetstore.unity.com>

<https://opengameart.org>

1. **Glossary & References**

Our team will use the following websites to use as game art, diagrams, and video tutorials to help us create our project:

<https://assetstore.unity.com>

<https://opengameart.org>

<https://www.youtube.com>

https://www.lucidchart.com/pages/

1. **Team Member Contributions**

Sebastian Farias (Team Leader and Technical Writer):

As team leader, I was tasked with the overview of the entire document, amongst other tasks. Firstly, I wrote section 1 and section 2 of the analysis document; this includes the introduction and a general overview of not only what the project will consist of but of the analysis document as well. I also created all of the diagrams that were used in the document, this includes the use case models, sequence diagrams, the activity diagram, and the class diagram, as well as their explanations. I created the models in order to give the team a general picture of what we had envisioned to include in our project. I was also tasked with outlining the user interface that we will implement for our game; this would include all menu options such as starting the game, displaying game instructions and credits, and the game art that we will use from the websites cited above in section 6. So in conclusion, my tasks were to outline and bring forth our envision that we as a team had discussed for our game “Birds of Prey”.

Maaz Ansari (Configuration Manager)

My principal contributions with the analysis document are the Use Case explanations for each part of the game. A use case serves as a description for the ways in which a user is able to interact with a system or product. The whole purpose behind a use case is to establish the requirements and create an outline in the many ways a user will interact with the system. A use case also manages the scope of things. Use cases are necessary because they can help communicate the strategy and how the game operates to others, especially to stakeholders. In a Use Case, there are multiple parts to it, such as the Use Case Name, the participating actors, the entry and exit conditions, the flow of events and any special requirements if they are present. For example, for the Use Case of Start Game, the participating actor was the player. The entry condition involved the Player choosing the “Start Game” option on the main menu. The exit condition was completing all the levels or selecting “Back to Menu” from the pause menu. The flow of events started from the Player selecting the “Start Game” button to ending off with the Player completing the level. If there is no Use Case, it is harder to communicate the way in which the user will be interacting the system which can create a problem in the creation and analysis of the game.

Anibal Ruiz (API-Liaison):

My principal contributions with the analysis document are the functional requirements and nonfunctional requirements. A functional requirement defines a system or its components. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Nonfunctional requirements define the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. For example, how fast does the system load? A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. I also contributed with the description or explanation of the Sequence diagrams, so the users have an idea how the program works when using any implementation.

Jean Claude Charles Boute (Webmaster):

My principal contribution with the analysis document is also part of the Nonfunctional requirements for the game, for example, the security of the game, which is a very important part of making the game. We want to make sure that the game is safe proof for everyone that is playing it. Our goal is to make a fun, safe game that everyone can play no matter the age since it is a general game, where some other games would need parental guidance. I also took some part of the user case descriptions, the point of the use case is to show what the actor or player will perform while playing the game. It is able to outline from the users point of view a very important perspective of the game that we are making. Use cases are represented as the steps that will be taken for example the goal of the game and how to fulfill these goals.