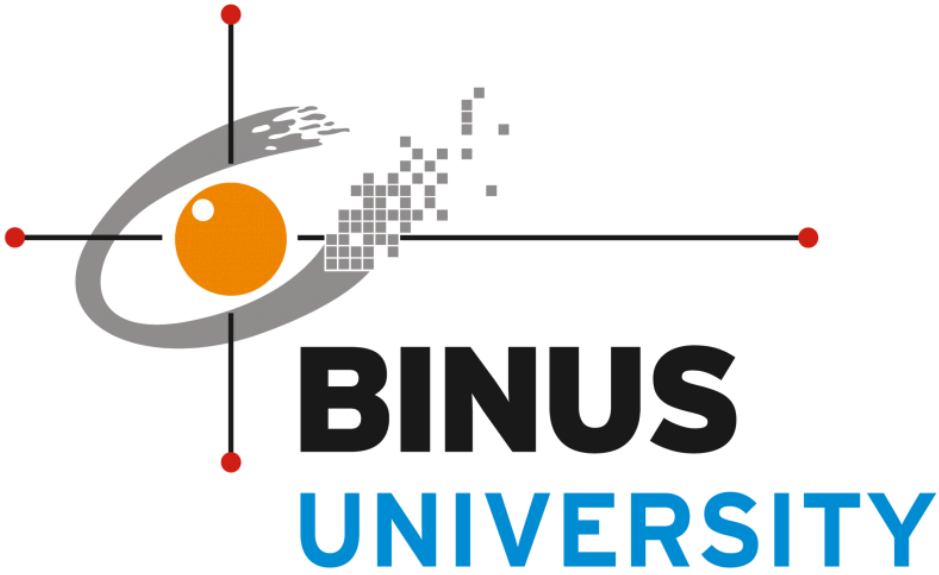
Final Project Algorithm and Programing

Collision Drift



Name: Ryan Alexander Kurniawan

ID: 2802530584

Class: L1CC

BINUS University International

Jakarta 2025

Table of Contents

Final Project Algorithm and Programing **1**

Table of Contents **2**

CHAPTER I INTRODUCTION **3**

Background 1.1 3

Inspiration 1.2 3

CHAPTER II DESIGN AND REQUIREMENTS **4**

Program Requirements 2.1 4

Program Design 2.2 4

Player Mechanics 2.3 4

CHAPTER III IMPLEMENTATION **5**

Use Case Diagram 3.1 5

Class Diagram 3.2 5

Activity Diagram 3.3 6

CHAPTER IV PROGRAM FUNCTIONALITY **7**

Github Link 4.1 7

Design Showcase 4.2 7

Extensibility of Program 4.3 8

Video Demo 4.4 8

CHAPTER V EVALUATION **9**

Suggested Improvements 5.1 9

Learned Lessons 5.2 9

Program Improvements 5.3 9

REFERENCES **10**

CHAPTER I INTRODUCTION

Background 1.1

This project is heavily inspired by my general love of games, and the era of old arcade games during the 1970s. Being a 2D top looking game with some 3D elements due to a feature later discussed called sprite stacking, the game resembles space invaders in a sense, though there are no bullets, a simpler enemy type using color visuals to identify the foe. Waves signaled by the song playing in the background has the added effect of progression when playing said game.

Though this project is far from perfect, in my limited time given I have hope to bring some challenge and thrill through the game I have made. Being the very first ever time I have created a game I hope to further improve it as my knowledge grows and improve its core features with added clarity.

Inspiration 1.2

The game takes heavy inspiration from a couple sources, mainly, Space Invaders, Car Racing Sim, Nova Drift (A steam game), Just Shapes and Beats, and Geometry Dash. The inspiration from Space Invaders comes from the idea of spaceships itself, the player, a spaceship stuck in space avoiding asteroids in the shape of simple blocks. The mechanics of the player's movement resemble the movement of a car drifting, letting go of "W" will lead to the player drifting as if with a normal car engine when the gas pedal has been released. Nova Drift, a game released in August of 2024 has also inspired the shape and premise of the player character in its objective to get to the end of the song. Speaking of song, the song was an inspiration from the game Geometry Dash, to further add progression and an idea of levels during the gameplay loop.

Furthermore, inspiration from Just Shapes and Beats, a game released on May 31, 2018, is the primary catalyst for the enemy type of the game. Simplifying the enemies to blocks and rectangles has given opportunities for other features to be developed, and an easier time to calculate collisions between the player model and enemies.

As of now, there is only a singular level to the game, but I hope to further add more levels in the progression of this game. Additionally adding special movement for different ship types, enemy types, obstacles, and a better screen collision system.

CHAPTER II DESIGN AND REQUIREMENTS

Program Requirements 2.1

Requirements to run the program are as follows:

- Pygame: To display the python program, window, and menu screen
- Python: The basic building blocks of the pygame program
- Sys: Access system-specific parameters and functions within the python runtime environment
- Imported: Random, Time, os, and Math'

Program Design 2.2

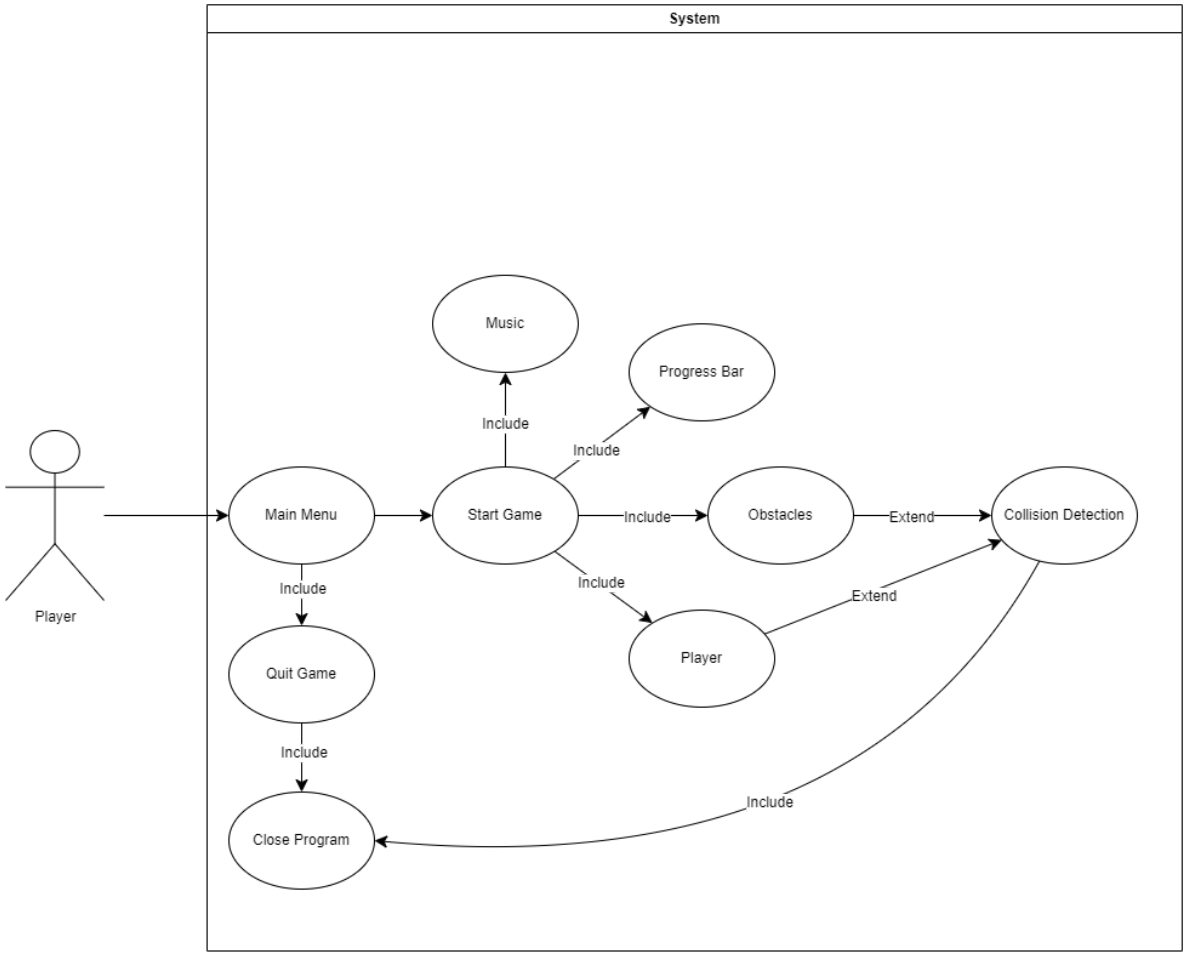
The program is designed to display a main menu with two buttons at the start of launch within "main.py", buttons are "Start Game" and "Quit". Pressing "Start Game" will lead you to the active game screen where the player character is spawned in, a progress bar is shown, music starts to play and the environment shifts to a space theme. After a second, the enemies, or red collision obstacles start to spawn, collision with any red object will result in the immediate termination of the program. Reason being are time constraints in creating a scoreboard, death screen or death animation during the progression of this game. The lack of priority of these minor features has led to the decision to terminate the program upon contact with any red obstacles. The song is roughly 3 minutes long, ending after the progress bar has been filled to the brim, as of which the program will also terminate.

Player Mechanics 2.3

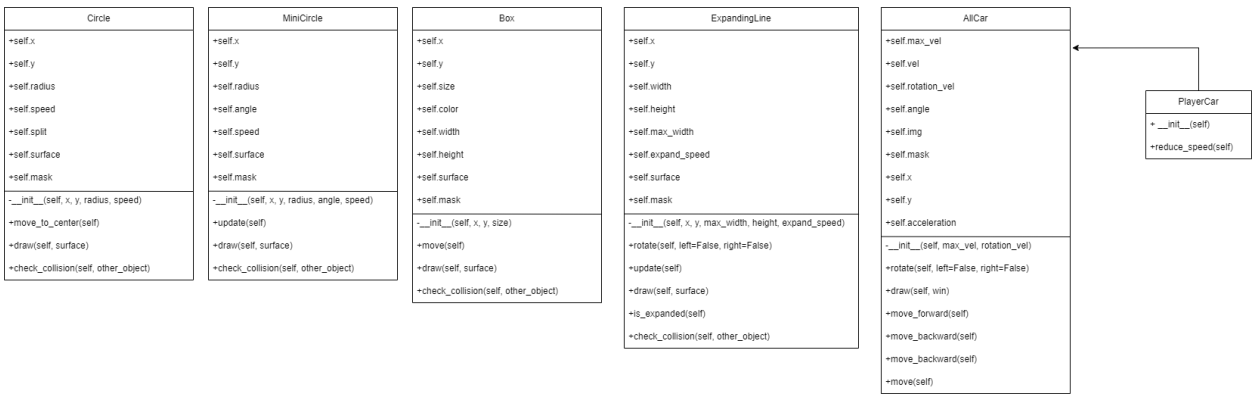
The player itself has two distinct unique mechanics tied to the movement and character model. To rotate the player model in the direction to be desired the function "blit_rotate_center" is called from "carutil.py", from the original image, the top left is equal to its own x and y value and the code makes sure that the center of that image is still centered on that point. Moving forwards has a slight tweak to it, as with car physics, pressing w and then letting go accelerates the player slowly and then after letting go, does not fully stop but drifts a few pixels just like a real car. Additionally, the sprite stacking visual is used to create a 3D feel with 2D assets in a 2D environment.

CHAPTER III IMPLEMENTATION

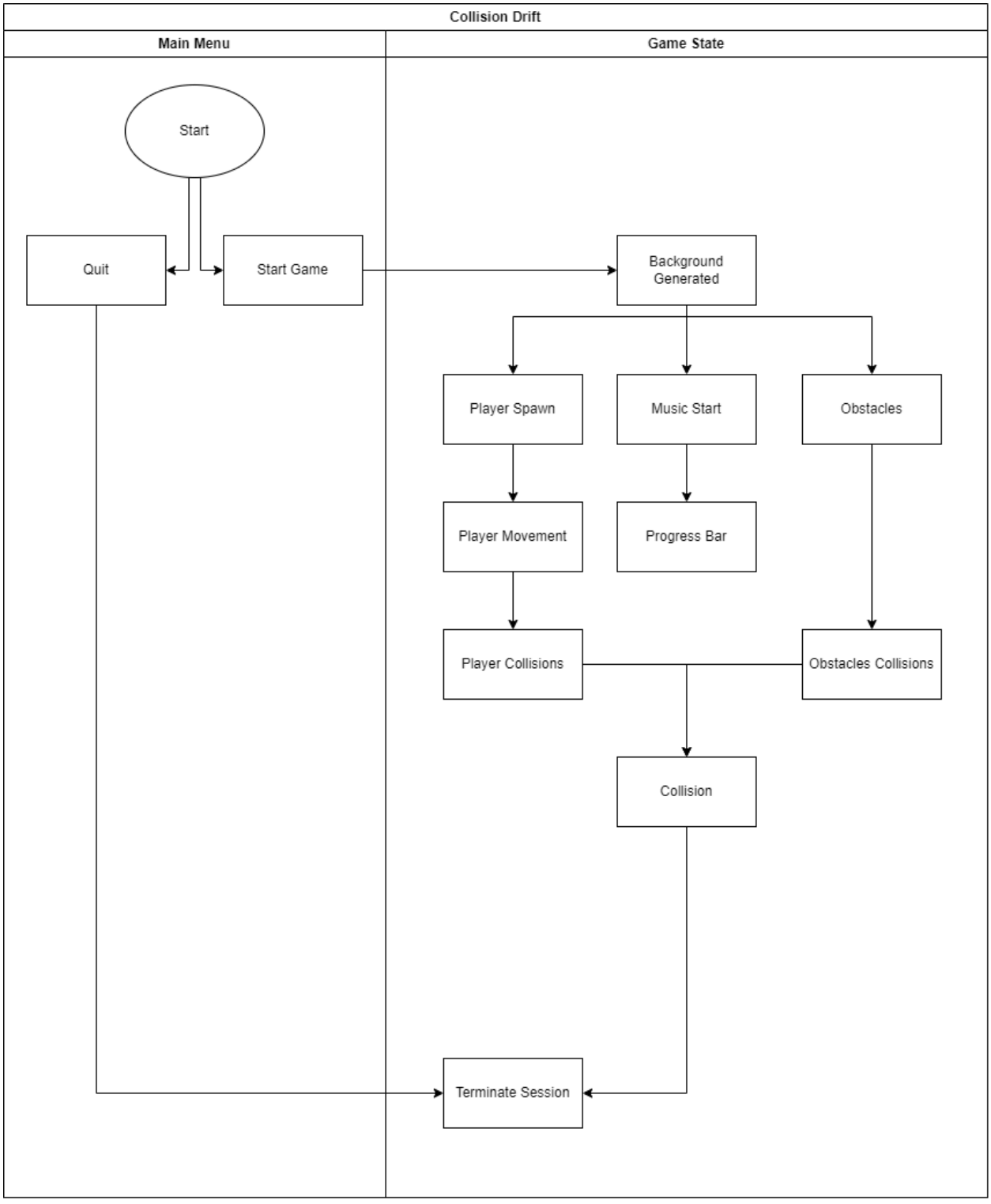
Use Case Diagram 3.1



Class Diagram 3.2



Activity Diagram 3.3



CHAPTER IV PROGRAM FUNCTIONALITY

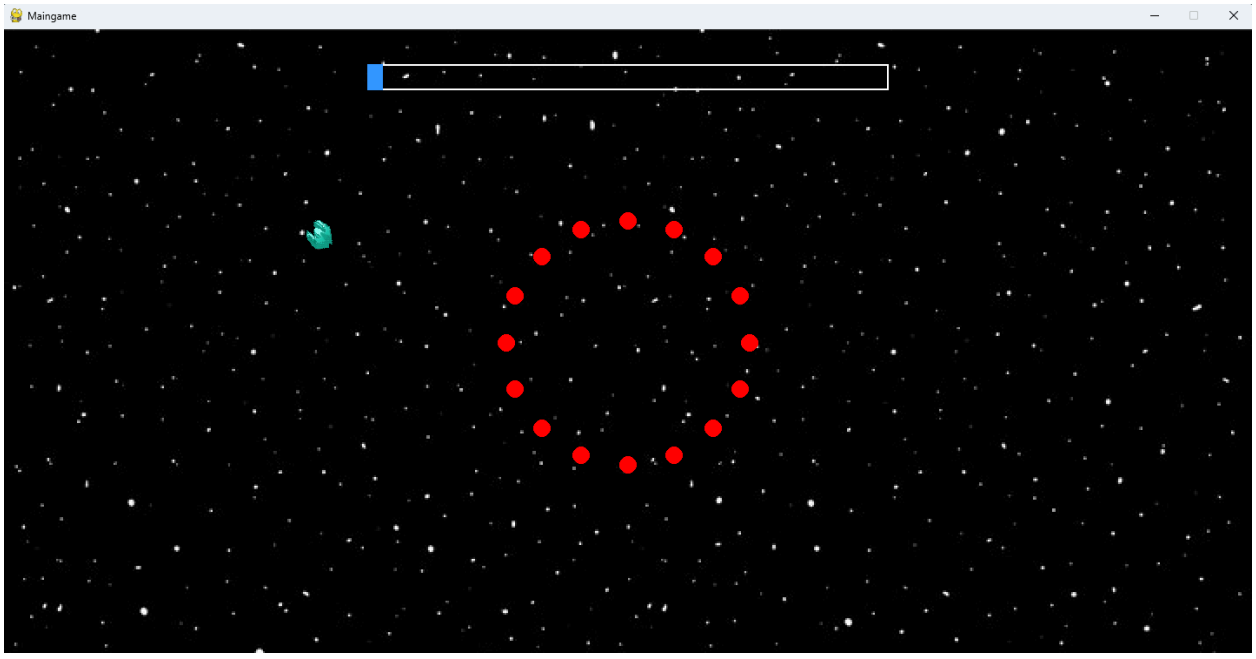
Github Link 4.1

https://github.com/FaultyDuck/AlgoprogFinalProject

Design Showcase 4.2

Figure 1. Main Menu

Figure 2. Game State



Extensibility of Program 4.3

1. Comment lines are present throughout the code to give further explanation to the functions of each line of code
2. Variables have unique names to identify the features and workings of each variable
3. Program structure is meaningful towards each state of the program

Video Demo 4.4

https://drive.google.com/drive/folders/1h-jV3cv2kGvx2XzB0f5roYpW_RFTXMyR?usp=drive_link

CHAPTER V EVALUATION

Suggested Improvements 5.1

Suggestions to improve the program's efficiency when running the game, remove the feature to terminate the program after hitting an obstacle, add a death menu, pause menu, further options in the main menu.

Learned Lessons 5.2

After the time it took to create this program there were many lessons that I have learned. I learned how to create instances of the same program without using external programs, to create a main menu to give the illusion of a flushed game. The journey taught me how to implement masks into the sprites used, creating my own sprites and a cool new way to apply a 3D effect on a 2D surface. I understand to appreciate smaller features in games more from this project, the features such as a pause menu and death screens are both quite difficult features to execute during this process. Overall, this game has taught me to be more resourceful with my time, priority of features, and what makes a good game challenging.

Program Improvements 5.3

- Create a death screen
- Create a pause feature
- Stop constantly checking and updating multiple collisions to avoid flickering in the gamestate
- Add more options to the main menu
- More levels
- More enemy types
- Fully pixel art enemies instead of blocks and circles only
- Checkpoint options for longer songs

REFERENCES

How to Use Pygame Masks for Pixel Perfect Collision | Coding With Russ [Online]
https://www.youtube.com/watch?v=tJiKYMQJnYg

Sprite stacking gamemaker 2d racing game that I'm making - fake 3d drifting | Loov Games [Online] https://www.youtube.com/watch?v=KKxlZ4cqfsk

Sprite Stacking in Python and Pygame | Coder Space [Online]
https://www.youtube.com/watch?v=HcRWqchSZOE

Spritestacks - Pygame Tutorial | DaFluffyPotato [Online]
https://www.youtube.com/watch?v=FNw_BQok14A

How to Create a Menu in Pygame | Coding With Russ [Online]
https://www.youtube.com/watch?v=2iyx8_elcYg

HOW TO MAKE A MENU SCREEN IN PYGAME! | Baraltech [Online]
https://www.youtube.com/watch?v=GMBqjxcKogA

Pygame Tutorial #6 - Enemies | Tech With Tim [Online]
https://www.youtube.com/watch?v=vc1pJ8XdZa0

Collisions in Pygame - Beginner Tutorial | Coding With Russ [Online]
https://www.youtube.com/watch?v=BHr9jxKithk