Modern War Simulator

Burak Şen - 150170063

Introduction

Defence strategy of nearly all tower defence games about placing your units in the game map in a predefined way. These games are played in a two-dimensional space and the strategies are based on just 2d space that is surrounded by them.

In this project, the enemy behaviour and defence units behaviours are going to be unpredictable to players. An enemy can decide to attack whichever defence unit it detects, and defend units will defend their position according to enemies behaviours. Users need to keep an eye on the 3d space to decide their strategy.

Methods

In this project, there are three primary methods: one algorithm from unit classes and two algorithms from MapGeneration classes.

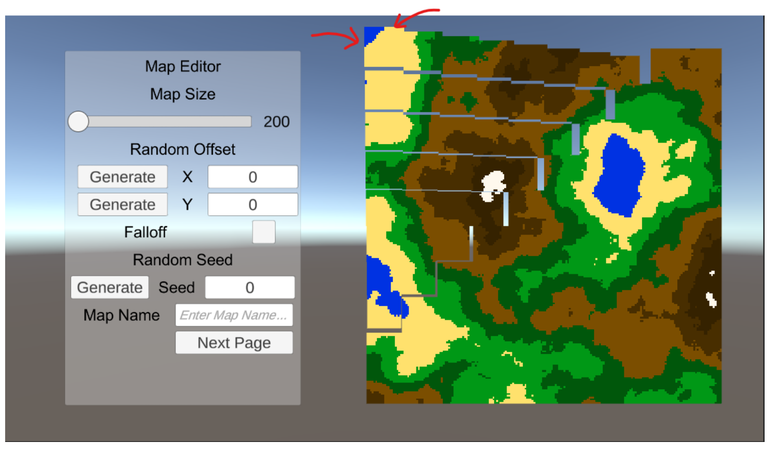
PERLIN NOISE ALGORITHM

The primary goal of this method is to generate randomized natural terrain. The game map's 2d texture and 3d model are produced using this approach. We must first develop a 2D heightMap texture in order to generate them.

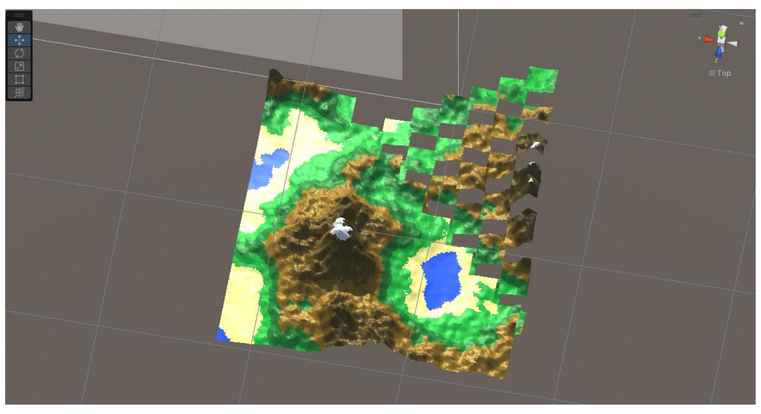
Output from Perlin Noise Algorithm - 2D HeightMap

MAP GENERATION ALGORITHM

The present noise technique makes it simple to create a noise map. However, as the size of the map we wish to produce grows larger, the application's speed suffers noticeably. Creating a 1000 by 1000 map was expensive in my tests. To solve this issue, we must divide our map into little sections and produce each component in its own thread. Fortunately, by offsetting each map by (size) *, our noise technique can build independent but related maps (index in their axis)



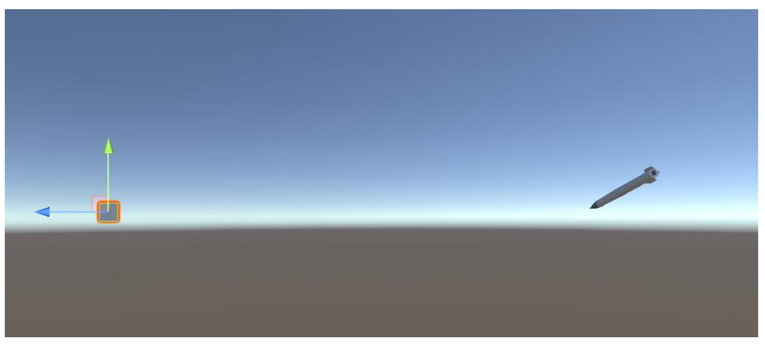
Map Editor Screen – 2D Map



Map Editor Screen – 3D Map

PID ALGORITHM – HOMMING MISSILE

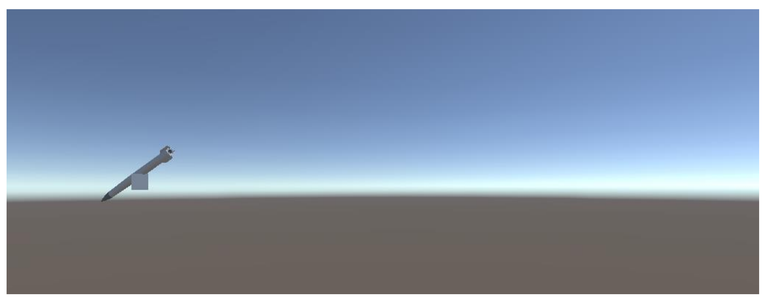
The main goal of this program is to generate a realistic missile firing. The PID algorithm was employed to obtain this behavior. This algorithm realistically applies whatever you desire. The p, I and d values are calculated in the PID Controller algorithm. The current value of the (set point) - (processed value), which is the current mistake, is proportional to the term "P." The term "I" calculates the "I" term by taking into account and integrating past values of the (set point) - (processed value) inaccuracy over time. Term "D" is the best forecast of the future trend of the (set point) - (processed value) error based on the current rate of change.



Pid - Algorithm - 1

In the Homing missile algorithm we just calculate each axis error value

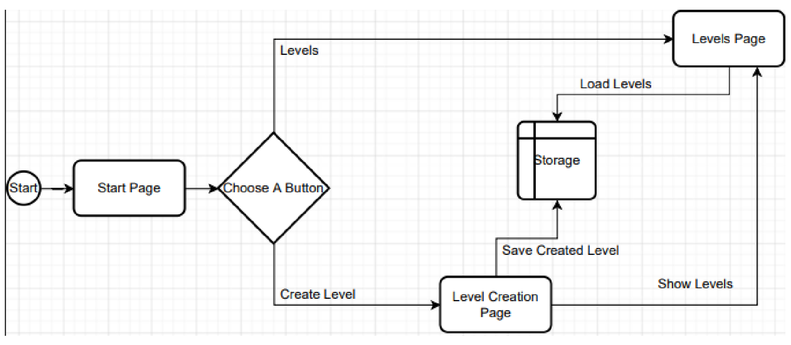
and according to these values, we generate a correction value and use this value as a torque in that direction.



Pid – Algorithm - 2

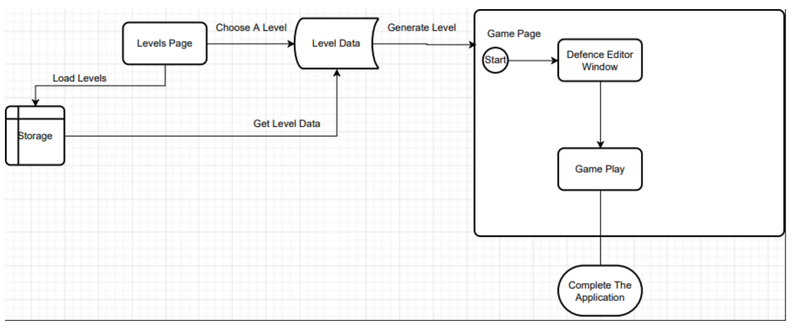
Dynamic Model

We either open a current level or create a new level on our level creation page in the start sequence. After setting the level choices on the level creation page, we can save our level, and it will appear on the levels page.



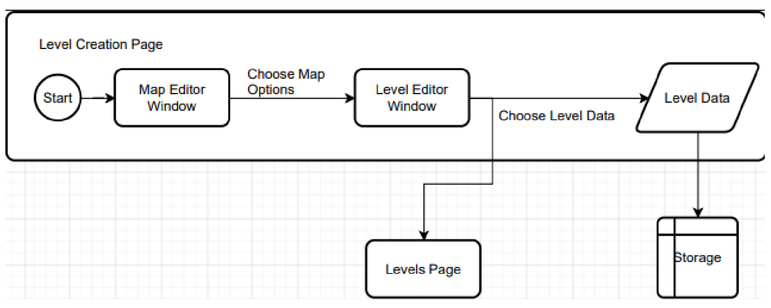
Start Sequence

We choose a game from the level page and load its data onto the game page during the gameplay session. Following the game, the page displays the defence editor window, prompting us to arrange our defense troops on the field. The simulation begins once we place our defense troops, and it continues until one side is vanquished.



Gameplay Sequence

The level editor process begins with the map editor window, which asks us to select several map parameters and displays the map modifications. After selecting the option, the level editor box appears, where we can choose the budget and attacker kinds. After that, we save the level to local storage and check it out on the levels page.

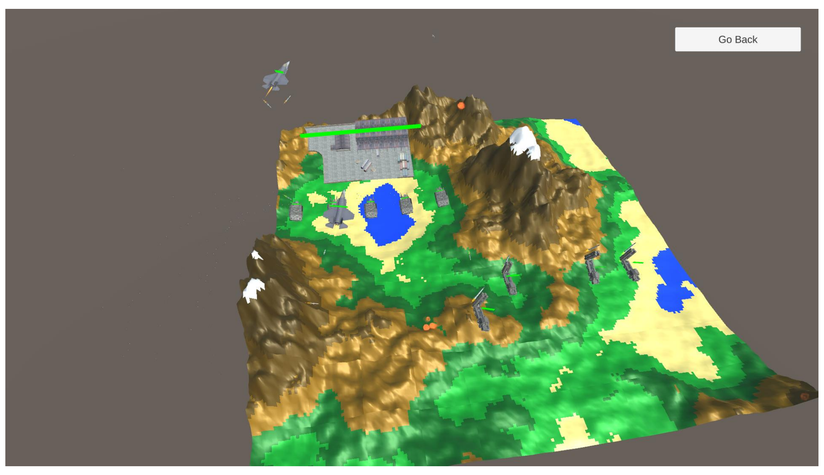


Level Editor Sequence

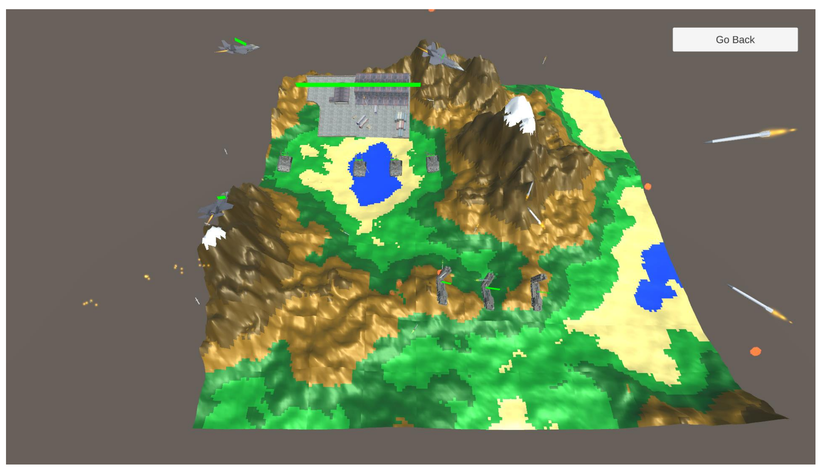
Conclusion

To conclude, this project covers a wide range of issues and use a number of methods to reach its goals. People are encouraged to utilize this project to show how to defend a three-dimensional space and to address problems they've encountered in this area.

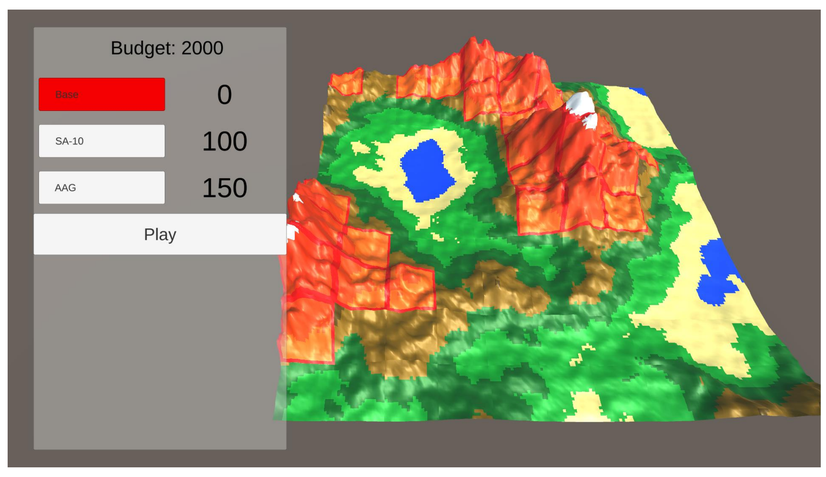
Another aspect of this project that should be noted is the engineering research. In this project, the unit system is the most important and adaptable framework. Each unit in the unit system is created and utilized using the Adapter and Abstract Factory patterns from the GoF patterns. There will be relatively minimal work to be done in the future if more units are needed for this project. Even if new features are need to be added to units in any way, doing so will be very straightforward due to the unit's structure and unit system, enabling for future improvements to be performed more swiftly.



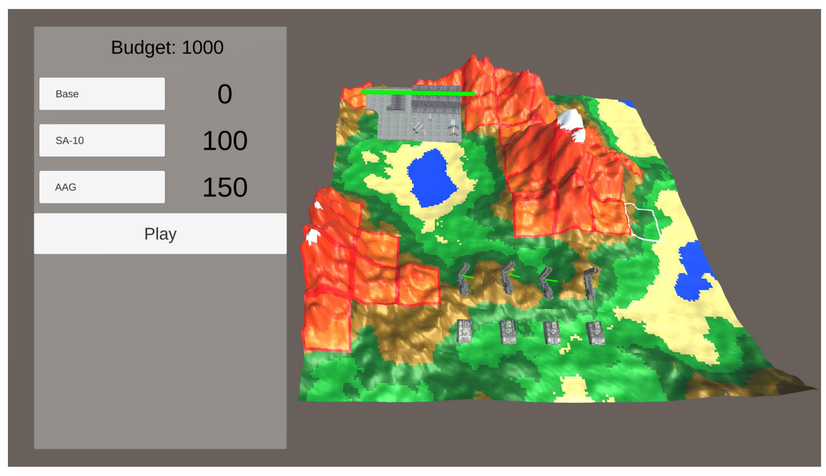
Gameplay-1



Gameplay-2



Gameplay-3



Gameplay-4

References

1. cadnav.com
2. cloud.anylogic.com
3. hash.ai
4. sebastian lague – youtube.com
5. Wikipedia.com – PID Controller

İTÜ
ISTANBUL TECHNICAL UNIVERSITY
Department of Computer Engineering

Faculty of Computer and Informatics Engineering
İTÜ Ayazaga Campus, 34469, Maslak/Istanbul
bbf@itu.edu.tr
http://bm.itu.edu.tr