Dragon:Unity

William George
Eric Hall
Michael Peters

Introduction

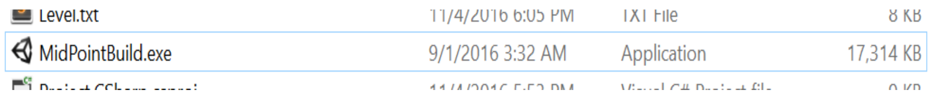
We've created a 2D tower defense game using the Unity Game Engine. The goal of the game is to keep the enemies from destroying the final base. Each member of the team worked on specific parts of the game to generate the current iteration. Specifically; Eric developed the menu screen and the navigation he also worked on adding a zombie enemy which will be in the final product, Michael developed/applied the a-star pathing algorithm for the enemy ai and worked on a golem enemy, and William developed a procedural map generator based on Conway's game of life to generate a random path each time a new game board is opened as well as developed a ninja enemy which will be added to the final product.

A description of what your program does from the user's perspective.

Our program is a top-down 2D tower defense game. The user clicks on the 'Next Wave' button to spawn a wave of enemies. The enemies will fly/move towards the ending point on the map. The user will be able to place towers throughout the map to stop the enemies from reaching or destroying the endpoint. The game is over when the enemies destroy the ending. Placing down a tower costs money. The user starts with a specific amount of money to use on towers. This adds in a strategic element to the gameplay, which makes it more interesting.

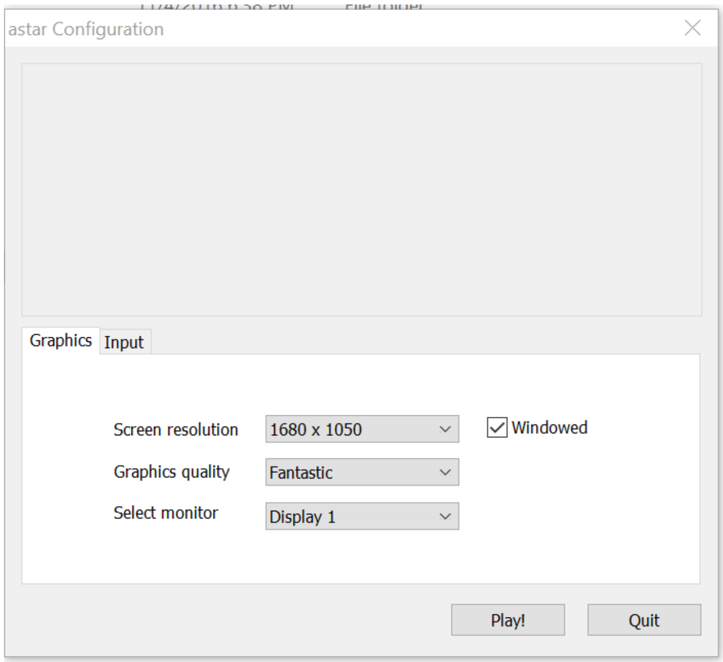
Usage Instructions:

1) Open the executable file MidPointBuild.exe



```
Level.txt          11/4/2016 6:05 PM    TXT File        8 KB
MidPointBuild.exe  9/1/2016 3:32 AM     Application    17,314 KB
```

2) Select your resolution, your graphics quality, screen size and make sure the Windowed box is checked

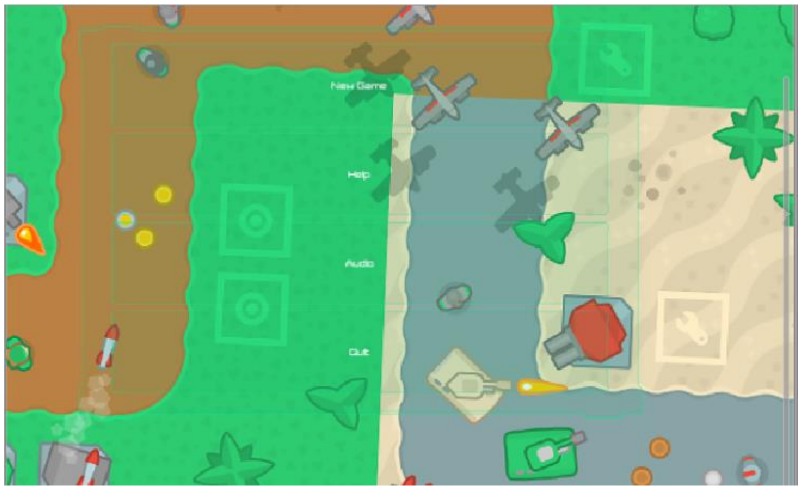


Screen resolution: 1680 x 1050 [✓ Windowed]

Graphics quality: Fantastic

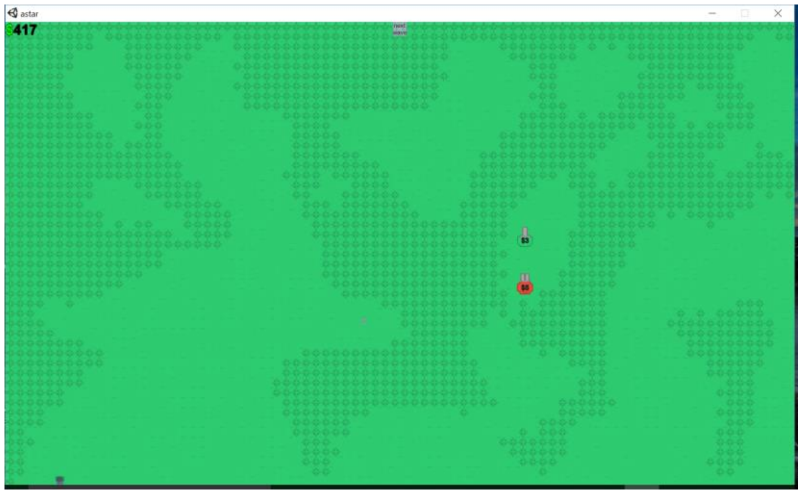
Select monitor: Display 1

3) Press the Play Button

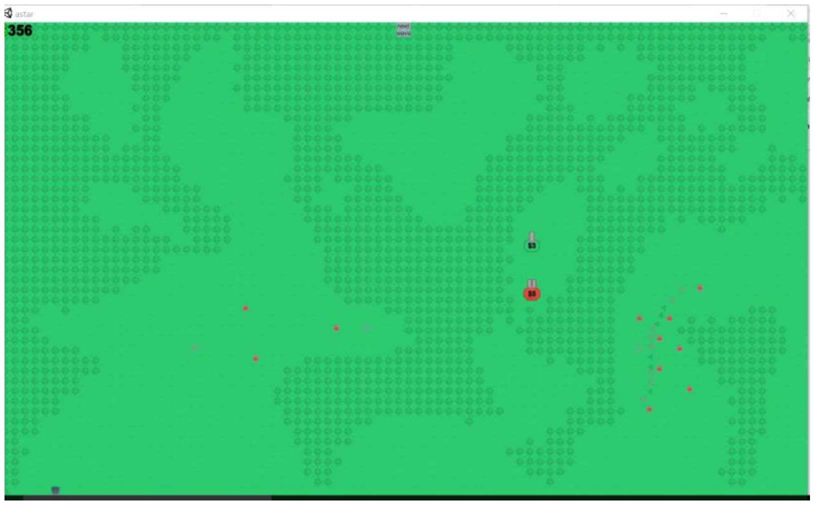


4) After the splash screen the game menu will open
5) Click New Game to start new game.
6) A randomly generated map with the following attributes:
    a) A Beginning Spawn point in the bottom left corner
    b) An Ending point in the top right corner
        i) This may be off the visible map (also the camera movement keys may not work to see the end spawn point)
    c) A starting money graphic in the top left corner

d) Two tank/tower graphics in the middle of the screen with the prices $5 and $3 on each
e) A next wave button located in the middle of the screen at the very top
f) The trees on the map indicate the blocked areas and the green grass sections show the usable path



7) To spawn a new wave of enemies, click "next wave."
   a) If the beginning spawn point is on a blocked area the path algorithm will not work, also if the ending point is on a blocked area or is unreachable the program will not work this is a bug we'll need to adjust in the second half of the project.
8) To place towers, click on a tower and then click on the screen where you would



like to place it.
   a) When you place a tower you'll notice the money decreases by the value of the tower

b) Also the tile where the tower is placed will not be available to place another tower
c) If an enemy is spawned before the tower is placed it will not block the path of the empty, otherwise it will

How the software and systems function together:

We have been using Unity 5.4. The Game file is a .exe executable and will work on any PC

Listing of software libraries, languages, APIs, development tools, servers, and other systems used to create the software:

- Unity API
    - Imported Assets:
        - Audio, Image, Animations/Meshes or other outside files types that can be used by unity
    - Asset Packages:
        - Packages of assets from items in the Unity Asset Store including Sprites and Backgrounds
    - Standard Assets
        - Collection of Assets commonly used by Unity Users
    - Scenes
        - Main Menu and Individual Levels as well as High Score/Game Over Screens
    - Game Objects
        - Enemies, weapons/defense towers, camera, background and the components and scripts of each defining individual behaviors/animations.
        - Scripts for the components of GameObjects will be written in C#
- The software will be built to be played by the user using standard input i.e. mouse clicks and keyboard inputs and released to work on PC, Mac & Linux as a Standalone game.

A description of what each Team member accomplished:

Week 3
Mike – Add Golem animation. Studying Unity.

William – Add Ninja animation. Studying how to incorporate Game of Life code to generate dynamic scene.
Eric – Added scene and studying unity

Week 4
Mike – Enemy spawn in different locations
William – Randomly generated game board
Eric – Zombie animation. Change states when button is pressed.

Week 5
Mike – Place towers on map. Button making.
William – Procedural map generator
Eric – Main Menu

Week 6
Mike – Astar Algorithm
William – More on procedural map generator, Combining Procedural Map Generator with AStar Algorithm and ensuring gameplay works.
Eric – Combining project elements. Transition from Main Menu to gameplay, handling gameplay navigation and working on midpoint documentation.

Current Bugs:

1) **Camera Navigation controls are not working properly**
2) **If a spawn point or end point is not on the grass path program will crash due to an exception**

The Next Steps:

Over the next couple weeks we are planning on doing the following actions to deliver a robust/bug-free game on time:

Week 7:
- Fix current spawn point, end point and camera bugs
- Refactor midpoint code
- Adjust camera settings so the map is easier to follow
- add new enemies to game map
- Add animation to tower units to track enemies and fire at enemies

Week 8:
- Add enemy and ending health point algorithms
- Add/call enemy death animations

- Add rules for completing the level and move to next level
- Add high score tracker

Week 9:

- Add skill/speed settings
- Add game play settings such as save game setting, so player can return to game
- Clean up game play

Week 10:

- Clean up loose ends, refactor code and test game play for bugs

Closing

` Before starting this project, none of us had any experience with Unity. Over the last few weeks each of us has become more acclimated to it. We have learned, as much as we could in the time span given, how Unity interacts with C#. None of us are experts in Unity yet, but we are more knowledgeable now than what we were before we started. We have learned in Unity there are game objects and each game object has a component attached to it. A game object can be anything… an enemy, weapon, level, etc. A component helps modify a game object. A component can be modified either directly in Unity or it can be done using a script. In Unity there are many built in classes, variables, constructors, public and static functions, as well as inherited members. We worked on understanding these concepts so we could have a better idea on how to approach this project. We are satisfied with the progress we have made so far and look forward to completing this project as a cohesive group.