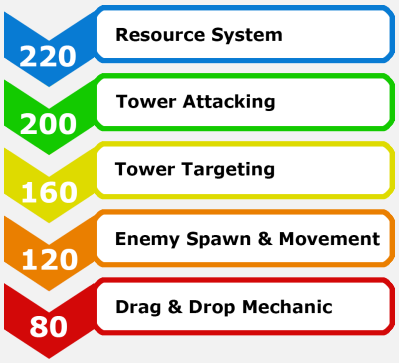
COMP1501A - Tutorial 07-08 Specification

Welcome to COMP 1501 - Introduction to Computer Game Design Tutorials. As a whole, the course's set of Tutorials are designed to be complementary to the material learned in class and assignments, and will emphasize, for the second half of the course, how to build a game in a team setting. To this end, this seventh Tutorial is the second of four designed with 'building a complete game' in mind.



Topics Covered This Week

- ☐ Placing Semi-Permanent Objects
- ☐ Auto-Targeting
- ☐ Spawning and Path Following
- ☐ Attacking Behavior
- ☐ Resource Generation

In this Tutorial, you will start the process of designing your Tower Defense game. To help you and save some time, a basic skeleton code has been provided as a base. You may opt to not use this however if you so choose. Grading for this Tutorial will be spread across two Tutorials, and you will have both Tutorials to complete these tasks.

Marks for this Tutorial will be based on completing certain milestones, where a failure to reach the first milestone will result in a grade of 0. You are expected to complete this Tutorial on your own time if you are unable to complete this during Tutorials. If you complete the Tutorial and demo it to a TA prior to the start of the subsequent Tutorial, you may earn half marks on the portion you completed outside of Tutorial time. It should be noted though that this is only an option if you were present for the Tutorial in question or if you've spoken with and gotten the approval of either the instructor or one of the TA's responsible.

Tutorial Submission

At the end of each Tutorial, please make sure to demo your work to a TA - this is when we will be live marking your submission. You are also expected to submit your work, zipped when there are multiple files, to CuLearn at the end of each Tutorial. Please also include, in a comment, the First Name, Last Name, and Student Number of each group member at the top of the Python Script.

lastname_1_lastname_2_tutorial07-08.zip

Introduction

This represents part 1 of 2 for a two-part Tutorial on coding up the base game mechanics for your proposed TD game. You will be given larger milestones, wherein each will be worth twice as much as they normally would be. Basically, rather than aiming for the 100% Tutorial, you will be aiming for the 200%. Whatever grade you receive overall (out of 200) will be evenly divided across the 2 Tutorials.

The following tasks are what you must complete (you may choose an alternate ordering if you wish). The first task is worth 80%, then each subsequent task completed is worth 40% apiece. The tasks are:

1. Drag and Drop Mechanics for placing towers.
2. Enemy Spawning and Movement.
3. Tower Targeting Mechanic.
4. Tower Attacking Mechanic.
5. Resource Generation and Usage.

In addition to these core mechanics to implement, you must also replace the 'sprites' and names within the Tower Defense game code to reflect your personal game design. This includes: map tiles, tower sprites, enemy sprites, enemy names, and tower names. Of course, you may add more to suit your game as well. If your proposed TD does not require one of the above listed mechanics, you may contact a TA and propose an alternative that you will implement. This alternative should be of an equivalent level of difficulty to the aspect you wish to replace. This mechanic will then be considered a 'core' mechanic and will not count towards a unique mechanic.

As a reminder, the following section re-describes the intent of your 'employer'.

Game Genre - Tower Defenses

Your employer, *Generic Game Company (GGC)*, has tasked you (your team) with creating a Tower Defense (TD) game. They strongly believe that this genre will be the best fit for showcasing a variety of mechanics and techniques. For the most part, they wish to provide your team creative freedom with regards to themes, story, etc., but they do wish to note several key requirements that you game MUST satisfy. These requirements are listed below:

Minimum Requirements

1. Enemies must spawn and must be attempting to go from one location to another.
2. The player must be able to make use of towers to defeat or reroute the incoming enemies.
3. There must be some sort of resource system in place to limit the player from immediately getting everything available.
4. There must be some way of losing the game and winning the game.

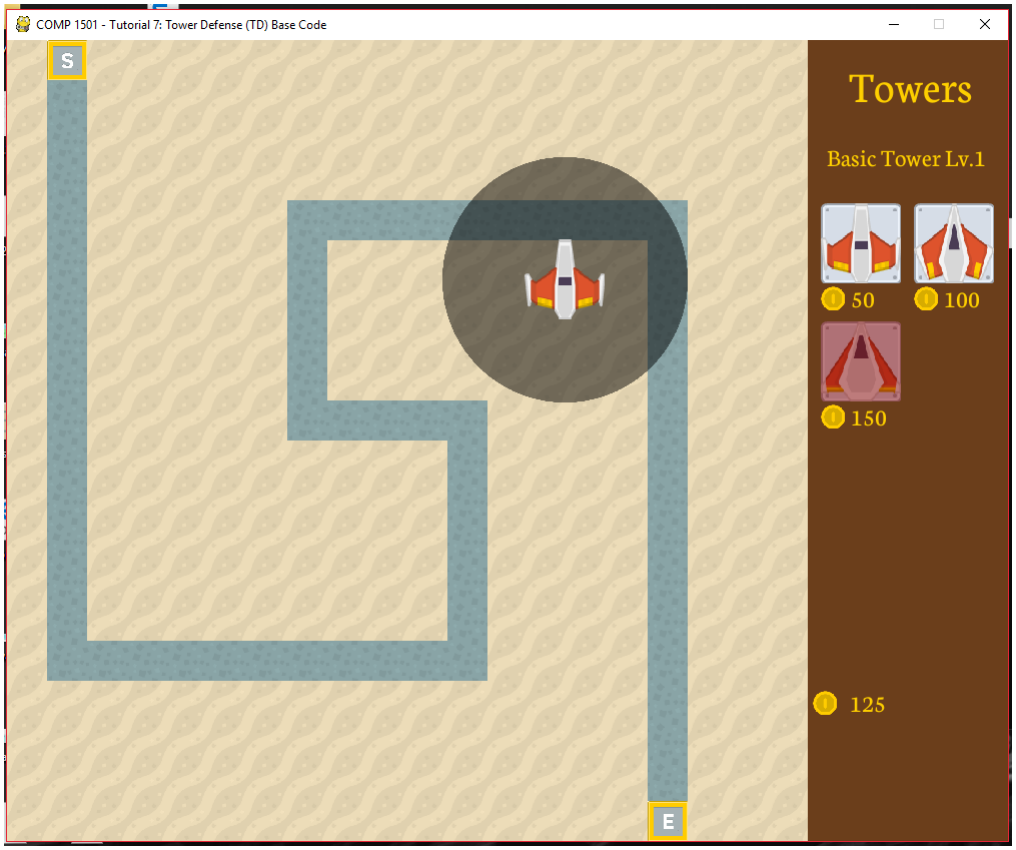
These should be considered the absolute baseline or minimum that your game should require. Your employer also notes several other key details or design decisions that you should consider (you are not required to directly answer these, just keep them in mind when designing your game):

Additional Considerations

1. How do the enemies travel across the map? Maze? Winding path?
2. How do you progress towards losing? Losing health? Lives?
3. What is your resources and how to do acquire it/them?
4. How are towers placed? Are there limitations on where they can be placed? Limitations on number of towers?
5. Which elements of your TD will be 'Real-Time'? Are towers placed during each round/when enemies are spawning, or is there a dedicated 'tower-placement' phase?
6. How do you handle the difficulty progression? Are towers upgradeable? More towers get unlocked?

Task 1: Drag and Drop Mechanic

This mechanic is a fairly standard one for TD games. You must implement the ability for a user to drag a tower icon from the UI onto the map, then be able to 'drop' said tower, which then gets placed on said location.



In more detail, the behavior of this can be described as following:

1. Check if the left mouse button is 'down'
2. If button is down, check if mouse position coincides with a 'tower' icon in the 'shop'
3. Check to make sure 'resource' cost is met
4. Draw tower at position of cursor along with a circle to indicate attack range
5. If mouse is moved, 'move' tower with mouse so long as left mouse button is still down
6. When left mouse button is released, check if mouse position is on map (not shop)
7. If mouse position is on map, check if location to place tower is valid
8. If valid, 'place' the tower in said location

Task 2: Enemy Spawning and Movement

In most instances of TD games, enemies will spawn from some set location or locations and move along some predetermined pathway towards your base. You must implement this behavior for your game. At minimum, this must include:

1. Enemies must spawn from some location
2. Spawned enemies must follow some path
3. Path must include some curve or turn

Task 3: Tower Targeting Mechanic

The next mechanic that you must implement is the tower targeting mechanic. This involves having towers scan the environment within their attack radii and then drawing a line connecting the tower to the enemy. You may of course decide how the target decides which enemy to target (i.e. first on the path, cloest to tower, last on the path, etc.). This line must follow the enemy as they move along the pathway.

Task 4: Tower Attacking Mechanic

Towers attacking is a very core aspect to the Tower Defense games. Your next task is to write the code to ensure that you towers can attack enemies within their respective ranges. This involves:

1. Checking if a tower has an enemy within range (see 'Tower Targeting Mechanic')
2. Check if it's 'time' to attack (based on tower's attack speed)
3. If time, fire an attack (i.e. bullet, arrow, laser, etc.)
4. When projectile intersects enemy, deal damage and destroy projectile
5. If enemy reaches 0 HP, destroy enemy

Task 5 (Bonus): Resource System

The final mechanic to implement is the resource system for your game. There is already a rudimentary default system in place, but you must either supplement or replace it. At minimum, you must have:

1. Killed enemies must generate some resource (or you must propose some alternate form of resource generation)
2. Towers cost a certain amount of resource

You may (and are definitely encourage to) implement a more complex and complete resource system, including having multiple types of resources. However, to get the marks for this component, only a the above must be included. Also note that this section/task is worth a fixed 20% rather than the 80 or 40 percent of the previous sections.