Backlog

# Game overview

We want to make a tower defense game with multiple towers and enemies. We also want to randomly generate the path the enemies have to take and the places where the player can put towers.

# Necessary components

Below are the components that we want the final game to contain and the properties and that these components need to have.

## Game field:

* Needs to have a path from beginning to end
  + Randomly generated by generating points and then connecting them
* Needs to have spaces for the towers
  + Randomly generated by having one close to each generated path point and then a few more randomly generated
* Spaces can be clicked to choose/upgrade tower

## Towers:

* They need to be built/upgraded with gold
* They need to be able to shoot at a specific enemy
* They need to be able to choose and enemy to shoot at (the enemy most to the front)
* They need to have a range

## Gold:

* You get gold from killing monsters
* You spend gold to build/upgrade towers

## Enemies:

* They need to have health points
* They need to give gold when killed
* They need to move along a path
* They need to have a speed

## Game rules:

* Enemies need to be regularly spawned in waves
* The waves need to increase in difficulty
* When a certain number of enemies get to the end, you lose
* When you kill enough enemies, you win

# Priority:

Below are the things we need to do to create the game. These things are ordered in order of importance, meaning we work our way down to list when we are working on the game. In the list each thing is described by its name. If you click on an item from the list, it will redirect you to further below in this document where the *how to demo* and *notes* for each item can be found.

1. [Make game field, but not randomly generated](#_Make_game_field,)
   1. [Make a path](#_Make_a_path)
   2. [Make spots for the towers](#_Make_spots_for)
2. [Make one enemy](#_Make_one_enemy)
   1. [Give it HP and a speed](#_Give_it_HP)
   2. [Make it move along the path](#_Make_it_move)
3. [Make one tower](#_Make_one_tower)
   1. [Make sure it can be built on a space](#_Make_sure_it)
   2. [Give it a range](#_Give_it_a)
   3. [Make sure it can see the enemies in its range](#_Make_sure_it_1)
   4. [Make sure it can choose the enemy in its range that is furthest along the path](#_Make_sure_it_2)
   5. [Give it damage and a cooldown and make it shoot at the enemy](#_Give_it_damage)
      1. [Let the enemy receive the damage](#_Let_the_enemy)
      2. [Let the enemy check if it is dead](#_Let_the_enemy_1)
4. [Let the enemies be spawned in a wave](#_Let_the_enemies)
5. [Implement lose condition](#_Implement_lose_condition)
   1. [Count the number of enemies that made it to the end](#_Count_the_number)
   2. [Check after each enemy that reached the end if you lost](#_Check_after_each)
   3. [Make a lose screen](#_Make_a_lose)
6. [Implement win condition](#_Implement_win_condition)
   1. [Count the kills](#_Count_the_kills)
   2. [Check after each kill if you won](#_Check_after_each_1)
   3. [Make a win screen](#_Make_a_win)
7. [Implement gold system](#_Implement_gold_system)
   1. [Add gold to the total amount if the enemy dies](#_Add_this_gold)
   2. [Let the towers cost gold and they can only be built if you have enough gold](#_Let_the_towers)
   3. [Give the player an amount of gold to start with](#_Give_the_player)
   4. [Show the amount of gold on the screen](#_Show_the_amount)
8. [Create a start button that starts the game](#_Create_a_start)
9. [Make sure the game works now, because this is the minimal viable product. Save this version as it is and don’t change it or lose it.](#_Make_sure_the)
10. [Increase the difficulty of the waves gradually](#_Increase_the_spawn)
11. [Make a few other towers](#_Make_a_few)
    1. [Same steps as with previous tower](#_Same_steps_as)
12. [Make a few other enemies](#_Make_a_few_1)
    1. [Same steps as with previous enemy](#_Same_steps_as_1)
13. [Let the spawner be able to randomly choose an enemy to spawn](#_Let_the_spawner)
14. [Make sure the towers can be upgraded](#_Make_sure_the_2)
    1. [Determine a cost to upgrade the towers](#_Determine_a_cost)
    2. [Determine the added bonus damage and speed to the tower](#_Determine_the_added)
    3. [Make the tower slightly bigger after upgrading](#_Make_the_tower)
15. [Randomly generate the path](#_Randomly_generate_the)
    1. [Randomly generate the amount of points somewhere on the game field](#_Randomly_generate_the_1)
    2. [Connect the points and determine the path](#_Connect_the_points)
    3. [Make sure the enemies know the new path](#_Make_sure_the_3)
16. [Randomly generate the tower spaces](#_Randomly_generate_the_2)
    1. [Generate a tower space close to each path point](#_Generate_a_tower)
    2. [Randomly generate more points on the field and check if a tower space can be there](#_Randomly_generate_points)
       1. [It cannot be close to another tower space](#_It_cannot_be)
       2. [It cannot be on the path, but it needs to be close to the path](#_It_cannot_be_1)
    3. [Keep generating points until you have enough correct ones](#_Keep_generating_points)
    4. [Make each tower point a tower space where you can put towers.](#_Make_each_tower)
17. [The version you have now is what we wanted to achieve at the start](#_The_version_you)
18. [You can add more towers, more enemies, make the game look nicer, make the game play better, make the difficulty better or implement some other thing that makes the game better.](#_You_can_add)

## Item description

### Make game field, but not randomly generated

**How to demo:** When the application starts, we want to see a field with a path and some tower spaces. **Notes:** learning goals: rendering swing components, adding an image to a swing program.

### Make a path

**How to demo:** When the application starts, we want to see a field with a path on it. **Notes:** learning goals: rendering swing components, adding an image to a swing program.

### Make spots for the towers

**How to demo:** When the application starts, we want to see a field with tower spots on it.

### Make one enemy

**How to demo:** When the application starts, there is an enemy at the beginning of the path that moves along the path and disappears when it reaches the end. **Notes:** learning goals: creating a moving object in swing.

### Give it HP and a speed

**How to demo:** When the application starts, the enemy moves in a direction with a constant speed. **Notes:** learning goals: basic programming.

### Make it move along the path

**How to demo:** When the application starts, the enemy moves along the path. **Notes:** learning goals: changing direction of moving objects in swing

### Make one tower

**How to demo:** A tower can be built on a tower space and can shoot at an enemy. **Notes:** learning goals: interaction with a mouse in a swing program, interaction between two objects.

### Make sure it can be built on a space

**How to demo:** When a tower space is clicked on, a tower appears. **Notes:** learning goals: interaction with a mouse in a swing program.

### Give it a range

**How to demo:** When the tower is clicked on, you can see its range. **Notes:** learning goals: drawing (partly transparent) circles in swing.

### Make sure it can see the enemies in its range

**How to demo:** When the tower is clicked on, it prints the amount of enemies in its range. **Notes:** learning goals: interaction between two objects.

### Make sure it can choose the enemy in its range that is furthest along the path

**How to demo:** When the tower shoots (next point) is shoots at the furthest enemy.

### Give it damage and a cooldown and make it shoot at the enemy

**How to demo:** The tower regularly shoots at an enemy, which loses HP and is eventually killed and disappears. **Notes:** learning goals: drawing moving objects in swing.

### Let the enemy receive the damage

**How to demo:** When an enemy is hit by an arrow, it loses HP.

### Let the enemy check if it is dead

**How to demo:** When an enemy has no more HP left, it disappears. **Notes:** learning goals: removing objects.

### Let the enemies be spawned regularly

**How to demo:** When the application runs, a few enemies will spawn **Notes:** learning goals: creating objects and rendering them in swing.

### Implement lose condition

**How to demo:** When enough enemies made it to the end, a lose screen appears. **Notes:** learning goals: switching screens in swing.

### Count the number of enemies that made it to the end

**How to demo:** When an enemy made it to the end, print the total amount of enemies that made it to the end.

### Check after each enemy that reached the end if you lost

**How to demo:** When enough enemies made it to the end, the program prints that you lost.

### Make a lose screen

**How to demo:** When enough enemies made it to the end, you are taken to a screen that tells you that you lost. **Notes:** learning goals: switching screens in swing.

### Implement win condition

**How to demo:** When you killed enough enemies, the enemies stop spawning. When you kill the last remaining enemy, you are taken to a win screen.

### Count the kills

**How to demo:** When an enemy is killed, the total amount of kills is printed.

### Check after each kill if you won

**How to demo:** When you killed enough enemies, no more enemies are spawned and if the last enemy is killed, the program prints that you won.

### Make a win screen

**How to demo:** When the player wins, they are taken to a screen that tells them that they won. **Notes:** learning goals: switching screens in swing.

### Implement gold system

**How to demo:** When an enemy is killed, you get gold. When you build a tower, you lose gold. The total amount of gold is shown on the screen. **Notes:** learning goals: showing a variable on a swing program.

### Add gold to the total amount if the enemy dies

**How to demo:** When an enemy dies, the total amount of gold is printed.

### Let the towers cost gold and they can only be built if you have enough gold

**How to demo:** When you want to build a tower, you lose gold and if you do not have enough gold, you cannot build a tower.

### Give the player an amount of gold to start with

**How to demo:** When the application starts, the player already has gold.

### Show the amount of gold on the screen

**How to demo:** The total amount of gold the player has, is shown in a corner of the screen. **Notes:** learning goals: showing a variable in a swing program.

### Create a start button that starts the game

**How to demo:** When the application starts, the player is shown a button labeled “start”. When the player clicks on the button, the game starts. **Notes:** learning goals: creating buttons in swing, waiting until a button is clicked.

### Make sure the game works now, because this is the minimal viable product. Save this version as it is and don’t change it or lose it.

**How to demo:** The game can be played as intended, without (major) bugs. **Notes:** The minimum viable product is the smallest version of the game that can be played. We need to make sure we do not lose this, because if we screw up when adding more things, we need to have a stable version of the game to return to.

### Increase the spawn speed gradually

**How to demo:** Each wave has more or more difficult enemies than the last.

### Make a few other towers

**How to demo:** When the player clicks on a tower space, a menu opens up that lets the player choose a tower to build. Each tower can be built and has an attack or other function to help the player. **Notes:** learning goals: creating menus in swing.

### Same steps as with previous tower

**How to demo:** See the description of the steps from creating the first tower. **Notes:** we create new towers in the same way that we created the first tower. Some towers may have different abilities than the first tower, which may require a different approach.

### Make a few other enemies

**How to demo:** See the description of the first enemy. **Notes:** we create the new enemies in the same way that we created the first enemy. The only difference between the different enemies is their speed and HP, so creating more enemies shouldn’t be that different from creating one enemy.

### Same steps as with previous enemy

**How to demo:** See the description of the steps of the first enemy.

### Let the spawner be able to randomly choose an enemy to spawn

**How to demo:** When the program is running, different enemies are spawned at the start of the path. **Notes:** learning goals: choosing a random element from a list.

### Make sure the towers can be upgraded

**How to demo:** When the player clicks on a tower, they can choose to upgrade it. When they do this, the tower does more damage and shoots faster. It also gets slightly bigger. **Notes:** learning goals: showing buttons when a tower space is clicked, changing the appearance of objects in swing.

### Determine a cost to upgrade the towers

**How to demo:** When the tower is clicked on, it shows a button with the cost of upgrading on it. **Notes:** learning goals: showing buttons when a tower space is clicked.

### Determine the added bonus damage and speed to the tower

**How to demo:** When a tower is upgraded, it does more damage and shoots faster.

### Make the tower slightly bigger after upgrading

**How to demo:** When a tower is upgraded, it look slightly bigger. **Notes:** learning goals: changing the appearance of an object in swing.

### Randomly generate the path

**How to demo:** When the application starts, it shows a field with a randomly generated path. **Notes:** learning goals: generating random numbers, drawing a fluid line through points.

### Randomly generate an amount of points somewhere on the game field

**How to demo:** When the application is started, there are points on the game field. **Notes:** learning goals: generating random numbers.

### Connect the points and determine the path

**How to demo:** When the application is started, there is a fluid path on the game field. **Notes:** learning goals: constructing a fluid line through points, drawing a fluid line in swing.

### Make sure the enemies know the new path

**How to demo:** When an enemy is spawned, it follows the randomly generated path.

### Randomly generate the tower spaces

**How to demo:** When the application is started, there are some tower spaces close to the path. **Notes:** learning goals: randomly generating points.

### Generate a tower space close to each path point

**How to demo:** When the application is started, there are tower spaces close to each generated path point that the path goes through. **Notes:** learning goals: generating points close to other points.

### Randomly generate more points on the field and check if a tower space can be there

**How to demo:** When the application is started, there are tower spaces on good spots (close to the path, but not on the path, and too close to another tower space.)

### It cannot be close to another tower space

**How to demo:** None of the drawn tower spaces are too close to another tower space.

### It cannot be on the path, but it needs to be close to the path

**How to demo:** None of the drawn tower spaces are on the path or too far away from the path.

### Keep generating points until you have enough correct ones

**How to demo:** When the application starts, there is a good amount of tower spaces on the game field.

### Make each tower point a tower space where you can put towers.

**How to demo:** You can click on a tower space to build a working tower. **Notes:** learning goals: creating objects on specific places in swing.

### The version you have now is what we wanted to achieve at the start

**How to demo:** The game is running without (major) bugs and contains all the elements described before. **Notes:** The version we have now, is what we wanted to achieve at the start. We need to save this version and not lose it, so we always have a complete working game. If we have more time, we can add more features to the game to make it better.

### You can add more towers, more enemies, make the game look nicer, make the game play better, make the difficulty better or implement some other thing that makes the game better.

**How to demo:** The added feature works correctly in the game. **Notes:** If we have enough time after finishing the game, we can add more features to the game, or balance the difficulty of the game better, so it is more enjoyable to play. What we do depends on the amount of time we have left and what we thinks is the best way to move forward to improve the game.