CBL game development
Jesse Janssen (1992538)
Aniek Goeree (1974459)

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.

Learning goals

The two new things we want to learn during this project are version control (git) and creating algorithms that can generate random things.

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

CBL game development
Jesse Janssen (1992538)
Aniek Goeree (1974459)

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. Only the items most to the left are part of the backlog features, the rest are the way how to do it.

1. Make game field, but not randomly generated
    1. Make a path
    2. Make spots for the towers
2. Make one enemy
    1. Give it HP and a speed
    2. Make it move along the path
3. Make one tower
    1. Make sure it can be built on a space
    2. Give it a range
    3. Make sure it can see the enemies in its range
    4. Make sure it can choose the enemy in its range that is furthest along the path
    5. Give it damage and a cooldown and make it shoot at the enemy
        1. Let the enemy receive the damage
        2. Let the enemy check if it is dead
4. Let the enemies be spawned in a wave
5. Implement lose condition
    1. Count the number of enemies that made it to the end
    2. Check after each enemy that reached the end if you lost
    3. Make a lose screen
6. Implement win condition
    1. Count the kills
    2. Check after each kill if you won
    3. Make a win screen
7. Implement gold system
    1. Add gold to the total amount if the enemy dies
    2. Let the towers cost gold and they can only be built if you have enough gold
    3. Give the player an amount of gold to start with
    4. Show the amount of gold on the screen
8. Create a start button that starts the game
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.
10. Increase the difficulty of the waves gradually
11. Make a few other towers
    1. Same steps as with previous tower
12. Make a few other enemies
    1. Same steps as with previous enemy
13. Let the spawner be able to randomly choose an enemy to spawn
14. Make sure the towers can be upgraded
    1. Determine a cost to upgrade the towers
    2. Determine the added bonus damage and speed to the tower
    3. Make the tower slightly bigger after upgrading
15. Randomly generate the path

    1. Randomly generate the amount of points somewhere on the game field
    2. Connect the points and determine the path
    3. Make sure the enemies know the new path
16. Randomly generate the tower spaces
    1. Generate a tower space close to each path point
    2. Randomly generate more points on the field and check if a tower space can be there
        1. It cannot be close to another tower space
        2. It cannot be on the path, but it needs to be close to the path
    3. Keep generating points until you have enough correct ones
    4. Make each tower point a tower space where you can put towers.
17. The version you have now is what we wanted to achieve at the start
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.

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 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.

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, drawing (partly transparent) circles in swing, interaction between two objects, removing object in swing.

Let the enemies be spawned in a wave

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.

Implement win condition

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

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 in a swing program.

CBL game development
Jesse Janssen (1992538)
Aniek Goeree (1974459)

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 difficulty of the game 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.

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.

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.

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, creating an algorithm that generates a random 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, creating object 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.

CBL game development
Jesse Janssen (1992538)
Aniek Goeree (1974459)

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.