Planetary Defense

Group Memebers

Kyle Bryant (kcbryant@wpi.edu)

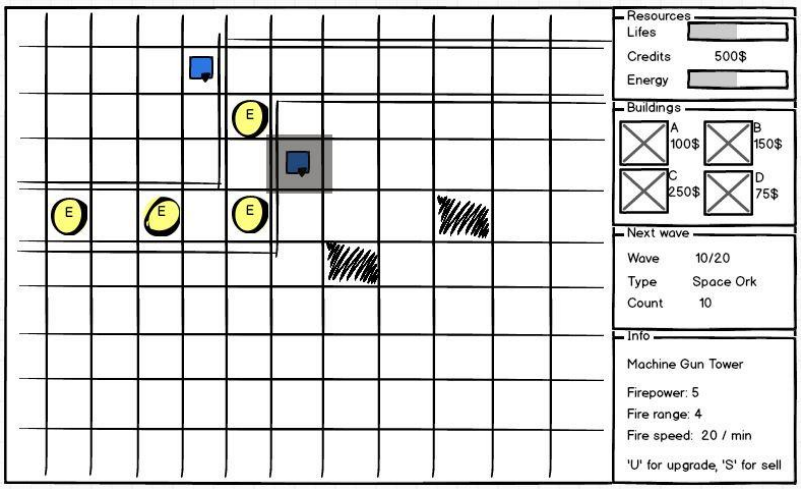
Benjamin Sautermeister (bsautermeister@wpi.edu)

Genre

Planetary Defense is a single player top-down tower defense game in space.

Description

Planetary Defense will be a top-down tower defense game for a single player. The game world will be based on a grid system with multiple cells. The player can navigate from cell to cell by pressing the arrow keys. Furthermore, the player can buy towers and place these on the currently selected cell using hotkeys, if the cell is not blocked. To give some variety, the player can choose between different towers with different properties like fire rate, firepower or fire range. To make the game more challenging, the play has to gain credits by destroying enemies and gain energy by building solar panels. Enemies will spawn one after another and try to make their way to the planet on a specific path. In addition, after every tenth level a single boss enemy with lots of armor must be defeated. This is essential to collect a huge amount of credits, which is required to master the next levels with less difficulty. It is in the player's best interest to set up towers in a way that prevents all enemies from reaching the planet. Each time an enemy reaches the earth, the player loses one of its life points. The game ends after the player has defeated the planet from all incoming enemy waves or when no life point is left anymore.



Technical Features

The following list gives an overview of the specific, significant technical features that the game will include:

- Map
  - Requires an MapManager or an extended ResourceManager that allows to load a map file
  - Defines the path, which the enemies are following by using an array of positions
  - Defines the active cells, which can be used by the player to build towers, but also blocked cells, which are used for the enemy path or background objects
- Light-weight grid system
  - Consists of multiple cells of the same size
    - Each cell holds information about its containing game object and whether it is active or blocked
  - Provides a virtual cursor, which allows navigation from cell to cell using the keyboard's arrow keys
- Enemies
  - Follow a given path
  - Have different properties, for instance speed, health and credits
  - Decrease one of the player's life points when they complete their whole path
  - The player gains credits, if he destroys enemies
- Bosses
  - A special kind of enemy, which usually has more health but a lower speed
  - Always appears alone after nine enemy waves
- Buildings
  - Towers
    - Have different costs, fire range, firepower and fire speed
    - Automatically fire to the first enemy which enters their sight
  - Supply
    - Solar panel to gain extra energy, which is required to build more towers

- Explosions
  - Very similar to the explosions of project 1
- Weapons
  - Used as a visual effect only
  - Flies from a tower to the enemy
- Events
  - EventKeyboard for cursor navigation and constructing towers
  - EventView to update the player's life points, credits and energy
  - EventStep to update each enemy path, targeting or firing bullets of the towers and update some sprites to specific frame indices according to the current state
  - EventCursorPositionChanged for updating the sidebar menu
- Level information
  - Contains an array of wave information and an associated map
  - Each wave information defines the enemy type and the number of enemies
  - Could be loaded from a level file or hard-coded
- Main menu
  - Allows starting the game, shows some brief instructions and allows setting difficulty or the used map, if these optional parts are implemented
- In-game sidebar menu
  - Displays the player's life points, credits and energy
  - Displays construction opportunities including its price or energy requirements
  - Displays some information of the currently selected cell, respectively its containing game object

Beside these technical features, the game could be extended with more features, if there is time left to implement them:

- Extended SceneGraph, which allows to get all enemies in a specific range
- Game balancing using the rock-paper-scissors principle
- Possibilities to upgrade existing towers
- Possibility to sell an existing tower
- Multiple difficulties and several levels or maps
- Even more towers, buildings, enemies, bosses or weapons

Could also require additional events like EventSplashDamage or EventElectroShock depending on the implemented types

- Special super weapons like a nuke or bombs
- Also requires additional events like EventNuke
- Game over screen for winning or losing

As a consequence of the technical features, there are some restrictions which must be followed by the player:

- The terminal windows must be set to a specific screen size, so that the sidebar menu and the game world can be appropriately displayed

- Because of the low resolution of a terminal window, the maps might be relatively small. One way to solve that problem could be to have a larger game world than the displayed window. But as a downside, this would mean than the player is not able to see the whole action at a look

Artistic Assets

The following list gives an overview about the significant artistic assets that the game will include. The assets dimensions will be probably 5 pixels width and 3 pixels height, like we plan to define our grid size to an adequate number of cells in the players view.

- We will create at least three sprites for the towers (8 frames to allow to facing to vertical, horizontal and diagonal)
    - Plasma Cannon
    - Machine Gun Tower
    - Tesla Tower
- We will create at least one sprite for the required buildings (2 frames)
    - Solar Panel
- We will create at least three sprites for enemy units (at least 4 frames for facing horizontal and vertical)
    - Space Goblin
    - Space Ork
    - Space Boss
- Main Menu (1-2 frames)
    - Shows the game logo, a brief instruction text including the controls and, if implemented, possibilities to choose a map or difficulty
- Sidebar Menu (1 frame)
    - Shows all information, like the buildable building, the resources and informations about the current selected object
- Background for Game Map (map type asset with 1 frame)
    - A sprite like map file is required, but will have its own format to define the enemy path and the active cells
- Explosions (6-8 frames)
    - The sprite files of project1 could be used for that, maybe with some slightly changes in their size

Implementation Plan

The implementation of the game's technical features are planned to implement in the following way. As a starter, the grid system and the loading of map are fundamental features of the whole tower defense game, so we will start with the implementation of these. Because the map will be defined in a dedicated file in a specific format, the game engine's ResourceManager must be extended to allow loading and parsing these files. Other game information content like wave definitions will be hard-

coded at first, but could also be refactored later to load these information from a file to allow players can define their own levels.

After a map with a controllable cursor is working, the derived game object classes could be implemented on tested on map. Next, enemy waves and interactions between game objects can be implemented, including explosions and weapons. Lastly, the games side bar menu must be added to visualize the current game state and the building opportunities.

During the development, it is acceptable to work with dummy sprite files, so the artistic stuff could be done after the alpha phase. Some sprite files could also be used from SaucerShot of project1, as an example the explosion sprite file.

After the implementation, the game should be balanced in some iterative steps to ensure that the game is not too hard or too easy to ensure the player will have a lot of motivation to continue playing our game.

Distribution of Work

For collaboration, the control version system GIT is used with on private remote repository hosted on BitBucket. Furthermore, the responsibilities of the project are separated as follows, but the distribution is just tentative and might change:

Benjamin

- Project setup
- Grid system implementation
- including loading of maps
- virtual cursor navigation
- Level and wave information
- In-game sidebar menu
- General game workflow

Kyle

- Main menu
- Building and enemy classes
- Event extensions
- Artistic assets
- Youtube video

Teamwork

- Brainstorming of game

- Plan document
- Design document
- Presentation
- Game balancing
- Testing and debugging

Schedule

The following list indicates all milestones, which we have defined to check our designing, development and testing progress:

- Milestone #1: Brainstorming and Refinement
  - Finish the overall plan for our game
- Milestone #2: Grid System Complete
  - Fully implemented grid with working cells
- Milestone #3: Basic Game Functionality
  - Working enemies and towers with a win/lose condition
- Milestone #4: Game Expansion
  - Add more enemies and towers, as well as implementing necessary game currencies
- Milestone #5: Menus, Art Assets, and Polish
  - Create main menu, sidebar menu, and art assets to flesh out Planetary Defense. Fix general errors and bugs.
- Milestone #6: Balancing, Debugging and Testing
  - Test the entire game and balance it so that the player has a lot of fun
- Milestone #7: Presentation and Youtube Video
  - Create youtube video and plan presentation
- Milestone #8: Future Development?

An overview of the planned dates is given in the following GANTT diagram: