Planetary Defense

a single-player top-down tower defense game in space

Benjamin Sautermeister
Kyle Bryant

10/13/2013

Table of contents

1. Group Memebers

Kyle Bryant (*kcbryant@wpi.edu*)

Benjamin Sautermeister (*bsautermeister@wpi.edu*)

2. Genre

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

3. Description

Planetary Defense is a top-down tower defense game for a single player. The game world is 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 or buildings 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 in waves 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.

4. Technical Features

The following list gives an overview of the specific, significant technical features that the final implemented game includes:

- Map and Levels
  - Extended ResourceManager that allows to load a maps or wave definitions (levels) from a 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 size 5x3 pixel

        - 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 such as speed, health and credits, spawn delay, which are defined in the level files
    - Decrease one of the player's life points when they complete their whole path by sending an appropriate event
    - The player gains credits, if he destroys enemies. This is also implemented by firing and catching the appropriate event
- Bosses
    - A special kind of enemy, which usually has more health but a lower speed
    - Always appears alone after nine enemy waves, but could be changed by adjusting the level files
- Buildings
    - Towers
        - Have different cost, fire range, firepower and fire speed
        - Automatically fire to the first enemy which enters their sight
        - Towers with splash damage fire an detonation event
        - Can be upgraded with a maximum level of 5
    - Supply
        - Solar panel to gain extra energy, which is required to build more towers
- Explosions
    - Are implemented like the ones of project 1
- Weapons
    - Flies form a tower to an enemy
    - Some weapons like the machine gun are fast and use the collision event
    - Other weapons like the grenade are slow and fire a detonation event when the target position is reached
- Events
    - EventKeyboard: for cursor navigation, constructing towers and menu navigation
    - EventInfo: to update the player's life points, credits and energy
    - EventBuidlingChanged: for purchasing, upgrading or selling buildings
    - EventDetonation: for weapons with splash damage
    - EventEnemyInvasion: to notify that an enemy reached the planet
    - EventEnemyKilled: to notify the player, the sidebar and the spawner that an enemy has been killed
    - EventPlayerKilled: to notify the main game controller that the player lost all his lifes
    - EventPlayerWin: to notify the main game controller that the player defeated all waves
    - EventSkipTime: to notify the spawner that the player is ready for the next wave
    - EventWaveInfo: to notify the sidebar with information about the next incoming wave

- Main menu
    - Allows starting the game, shows some brief instructions. In addition, a second screen allows setting the difficulty or the used planet. More technically, this means which level file and which map file should be loaded.
- In-game sidebar menu
    - Displays the player's life points, credits and energy
    - Displays construction opportunities including its price or energy requirements
    - Displays information of the currently selected cell, respectively its containing game object
    - Displays information about the next incoming wave
    - Highlights important fields with red, yellow or green

As a consequence of the technical features, there is the restriction that the game must be started in a appropriate sized window. We recommend that the game is started using the full screen mode for the best user experience.

5. Artistic Assets

The following list gives an overview about the significant artistic assets that the game includes. The assets dimension is in the most cases 5 pixels width and 3 pixels height, like we defined our cell size of the grid system.

- Five sprites for the buildings (8 frames to allow to facing to vertical, horizontal and diagonal)
    - Grenade Tower
    - Machine Gun Tower
    - Tesla Tower
    - Laser Tower
- We created one sprite for the required building (2 frames)
    - Solar Panel
- We created three sprites for enemy units (at least 4 frames for facing horizontal and vertical)
    - Space Goblin
    - Space Ork
    - Space Boss
- Menu Screens
    - Separated in several screens
- Three game maps (map type asset with 1 frame)
    - Defined in a own file format, which consists of the grid and cell size, the passable map and the single frame background map
    - The animated stars of the background are implemented quite like in project 1
- Effect sprites
    - For effects like explosions, flying bullets or electro shocks, we used simple sprites like the explosions or bullets of project 1
- Three levels for various difficulty

- Defined in an own file format, which consists of the number of waves. Also, it defines the number, type, number, health and delay between the enemies.

6. Implementation Details

For the implementation for our game, we used the game engine of Benjamin, because it included all features provided by the Book and was well tested using a unit testing framework.

7. Distribution of Work

For collaboration, the control version system GIT is used with on private remote repository hosted on BitBucket. The distribution of work did not change much over the course of the project; however there are some slight variations from the original plan:

Benjamin

- Project setup
- Game engine extensions
- Grid system implementation
  - including loading of maps and level information
  - virtual cursor navigation
- In-game sidebar menu
- General game workflow using event
- Weapon implementation
- Game promotion website and promotion logo (200x150px)

Kyle

- Main menu, control menu and planet menu
- Building fundamentals
- Enemy implementation
- Enemy and wave spawning
- Artistic assets
- YouTube video
- Game pausing for extended construction time

Teamwork

- Brainstorming of game
- Plan document
- Building implementation
- Design document
- Presentation
- Game balancing
- Testing and debugging

8. Schedule

This is the original schedule updated with additional details and completion times.

8.1. Milestone #1: Brainstorming and Refinement:

- Planned: Oct. 2, Finished: Oct. 2
- Decided on a tower defense game that would work well with Benjamin's EngineX game engine
- Created a Plan document outlining milestones, division of work, and general necessities
- Created a GIT repository for Project 3

8.1.1. Summary of Milestone #1:

Our group brainstormed and drafted ideas for our final project quickly so that the team could begin development as soon as possible. We decided on a tower defense game because of its ease of implementation into an ASCII-based game engine and the overall challenge it presented to us as designers. For this part, we got together in the campus center and hashed out ideas on scrap paper, then put the plan document online as a Google-Document so that we could both edit and change it in collaboration before submission.

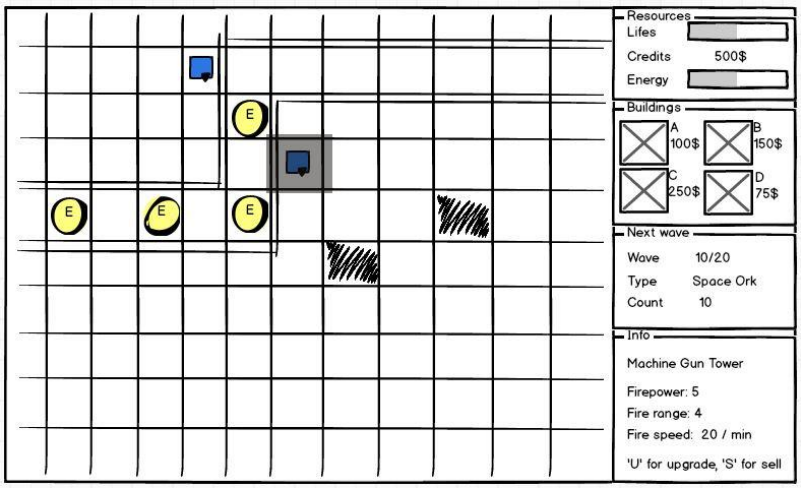

Figure 1 - Planed game

8.2. Milestone #2: Grid System Complete

- Planned: Oct. 4, Finished: Oct. 4
- Finished quickly so development of base game could continue quickly
- Grid and cursor completed, as well as loading of map files

8.2.1. Summary of Milestone #2:

Benjamin completed most of Milestone #2 before the weekend. This was planned for one of two reasons: We wanted a working base to have our game testable as soon as possible, and Benjamin

was going to be in Boston for the weekend. As a result, we decided that he would finish it up so that Kyle could work on the basic game functionality over the weekend while Benjamin was gone.

8.3. Milestone #3: Basic Game Functionality

- Planned: Oct. 10, Finished: Oct. 8
- Basic enemies and turrets added to game
- Enemy waves implemented

8.3.1. Summary of Milestone #3:

Kyle and Benjamin completed a large part of the game functionality before the Alpha week of the project. Kyle spent time over the weekend to implement a base class for enemies as well as creating the spawner and base class for towers. Benjamin implemented the necessary event classes needed for the game, the basic game workflow as well as a rudimentary sidebar. Our team created basic art assets as well to flesh out the testing.

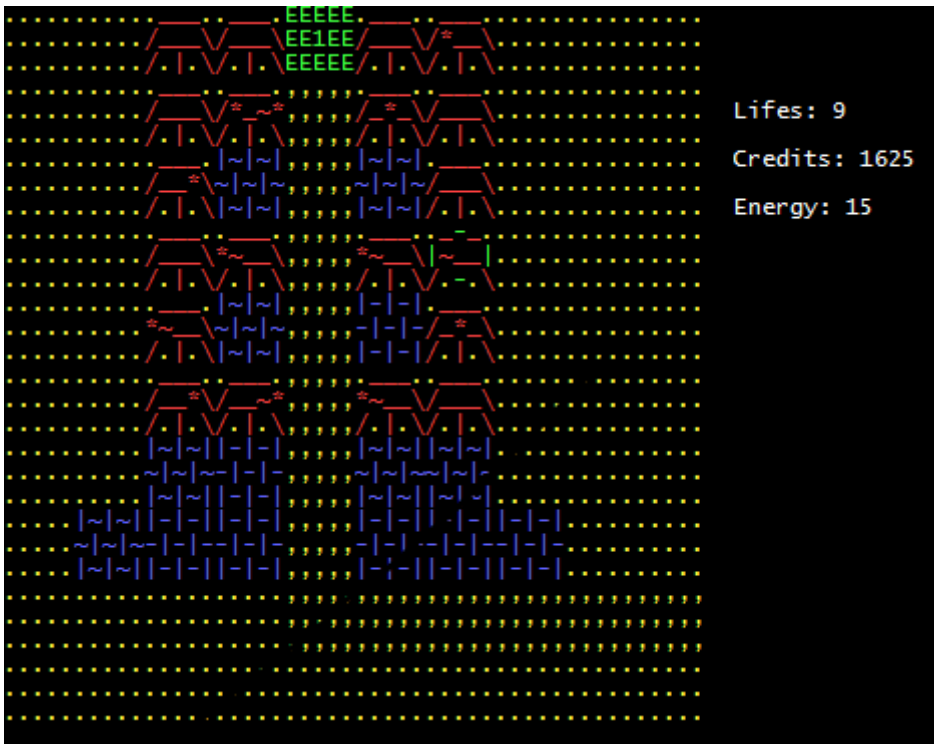

Figure 2 - Screenshot of game with basic game functionality

8.4. Milestone #4: Game Expansion

- Planned: Oct. 13, Finished: Oct. 10
- Added multiple enemy and tower types
- Added more sprites

8.4.1. Summary of Milestone #4:

Because of our rapid progress, much of the game expansion was completed for the alpha. Other than a last segmentation fault of the game engine, which occurred in the onEvent()-function of the manager base class, our game was in a playable and decent looking state.

8.5. Milestone #5: Menus, Art Assets, and Polish

- Planned: Oct. 14, Finished: Oct. 12
- Added main menu, controls, and sidebar menu
- Addressed Segmentation fault issues
- Added a win a loose scene

8.5.1. Summary of Milestone #5:

We wanted to get the game in release condition before the week the final was due. Kyle worked to add in artistic assets for the game menu, planet menu, and control menu. Benjamin worked diligently to find the pesky segmentation fault, as well as implementing a fully functioning sidebar with a beautiful UI. The fix of the segmentation fault took about three days and was supported by Prof. Claypool.

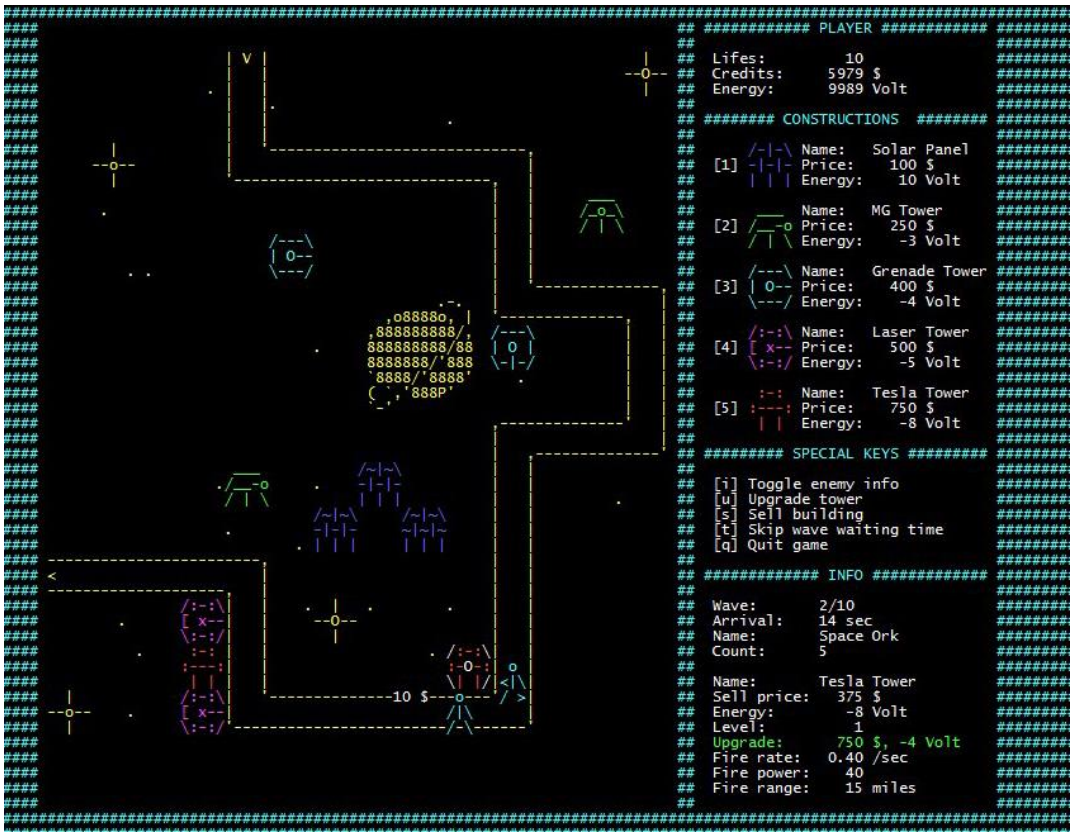

Figure 3 - Fully implemented game

8.6. Milestone #6: Balancing, Debugging and Testing

- Planned: Oct 15, Finished: Oct 13
- Tested game with another project team
- Balanced game for difficulty scaling

8.6.1. Summary of Milestone #6:

We worked with another team to white-box test out our games and find bugs, balancing or other possible issues we may have otherwise overlooked. We used some of their input to update our game and fix some balancing issues.

8.7. Milestone #7: Presentation and YouTube Video

- Planned: Oct 17, Finished: Oct 13
- Created a short video and uploaded to YouTube
- Wrote this design document
- Created game website
- Created other promotional material

8.7.1. Summary of Milestone #7:

Our team worked together to get all of the additional material done for Project 3. Kyle wrote the design document and created the YouTube video. Benjamin created a single page promotion website for the game, as well as a small promo image for Planetary Defense. The YouTube video and the promotion image is included in the created website.

9. Summary

Our team covered ground quickly and efficiently, communication was a key aspect in the development of Planetary Defense. Both team members took time to ensure Planetary Defense would live up to the image we had for it from the start. Our game was finished with full functionality that we had originally intended, as well as a few extras we had not originally planned to be a part of Planetary Defense (such as the Planet map) As you will have noticed by looking through the previous checkpoints, our team stayed relatively ahead of the original schedule, and the final product shows our hard work.