17\_0324 CSCI 313 Project 2 Game Specification – Ryan Nelson and Jack Morgan



**Game Title:**

Rocket Lander!

**Story / Narrative:**

Ever since humans first gazed at the stars in the sky, we’ve been fascinated with space. Though the field of space technology is very young, it has made incredible progress. It’s allowed us to create a global network of communication and sensing devices, send probes to distant planets, and put humans on the moon.

Today space technology is more important than ever. It allows us to learn more about our planet, more about the universe, and develop a way of becoming multi-planetary. However, the field is not progressing as fast as it could because it is very expensive to launch things into space.

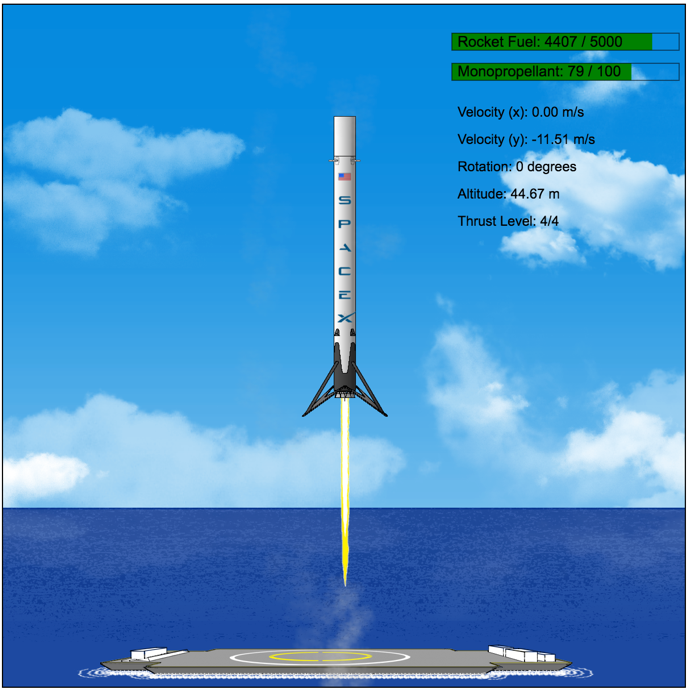
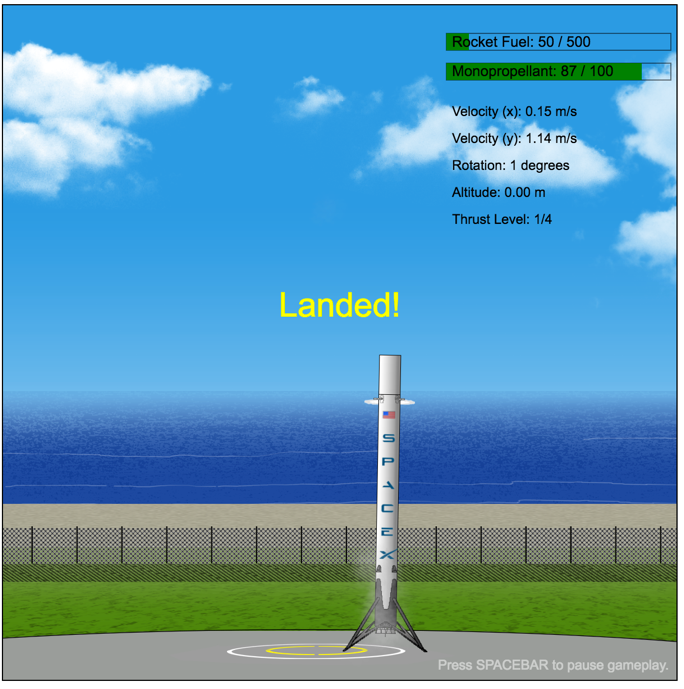
Rocket launches are expensive because rockets are not reusable. Consider this: if an airplane had to be thrown away after a single flight, the cost of flying would be enormous. Very few people would fly. So too with space.

One company, SpaceX, has been perfecting technology to land and reuse the first stage of its Falcon 9 rockets. This feat is extremely difficult, and after many failures, SpaceX has succeeded. It has now safely landed multiple first stages on land and sea sites and is preparing for reuse.

This game is a tribute to the incredible achievement of SpaceX. With Rocket Lander!, players get a chance to perform a critical part of the Falcon 9 first stage’s return to earth – the landing. Players will control a rocket modeled exactly on the Falcon 9 and will attempt to land it while being constrained by fuel and physics. This 2D game will have two levels – a land-based landing site, and an ocean-based landing site. Both sites will be closely modeled to resemble actual SpaceX facilities.

We also wanted to develop a game that challenges players but has no risk. Failed landings are spectacular events, but don’t penalize a player in future attempts or result in gameover. This allows the player to engage and enjoy the game without being stressed.

This game is different from our first game.



**Scene:**

The game is laid out in a single frame, with all content visible. The game background will be themed based on the level. The game’s GUI will include the following visualizations:

* remaining rocket fuel (for the main engines), monopropellant (for the thrusters)
* physics statistics (horizontal and vertical velocity, current angle, thrust level, altitude)

**Objects:**

Rocket: Players control the Falcon 9 first stage rocket.

* It has nitrogen cold-gas thrusters to change its orientation.
* It has Merlin engines to slow down the rocket for landing.
* It has grid fins at its top to help with orientation.
* The rocket has landing legs to absorb landing impact and stabilize the vertical mass after landing.
* Just like the actual rocket, the game’s rocket will use a single (central) engine for landing.
* Players control thrust level to vary the amount of force the rocket engine releases. High thrust uses more rocket fuel than low thrust does.

Landing Site: The landing site sets the location and altitude that a rocket can be landed.

* The width of the landing site varies depending on the level
* For the land-based landing site, the rocket can land anywhere on the concrete surface.
* For the ocean-based landing site, the rocket must land on the deck of the drone ship and not on its equipment or facilities.

**Physics**:

The game simulates the following physics phenomena:

* Gravity: The rocket falls to the surface at a continuously increasing rate.
* Momentum:
  + Horizontal: If engine thrust is stopped, the rocket continues to move horizontally in the direction it was traveling.
  + Vertical: It takes a certain amount of thrust to counteract the force of gravity and slow down the rocket. The rocket will not immediately fly upward when its engine is engaged, it must reduce downward velocity, reach zero, and finally can fly upward.

**Events**:

The game has the following events:

* Starting a new Attempt
  + At the start of an attempt, the rocket’s landing legs unfold
  + Within a range, the rocket spawns in a random horizontal position and angle
* Thruster Event
  + When a thruster is engaged, it results in a force that rotates the rocket
  + The thruster plume is also visualized and animated
  + Smoke is released from the end of the plume
* Engine Event
  + When the engine is engaged, it results in a force that controls vertical and horizontal movement
  + The engine plume is also visualized and animated
  + Smoke is released from the end of the plume
  + The force and animation are based on the current thrust level
* Successful landing
  + If the rocket lands successfully, the rocket stops moving and the GUI displays a notification
  + The rocket flares both of its thrusters, just like the actual rocket
  + The level is then restarted
* Failed landing
  + If the rocket fails to land, an explosion will be shown
  + The level is then restarted
* Changing levels
  + This event can only occur in between failed or successful landing events
  + The currently falling rocket is removed
  + This results in a new game background being shown, the landing site being updated, and restarts a new attempt for landing

**Rules and Objectives:**

Game

* The goal of the game is to land the rocket first stage. The following criteria for a “successful” landing are used:
  + Correct vertical position: rocket has the same vertical plane as the landing site
  + Correct horizontal position: the rocket and its landing legs are within the horizontal boundaries of the landing site. These boundaries vary depending on the level, and are not visible. Players must discover boundaries through multiple attempts.
  + Correct speed: rocket cannot be traveling faster than 2.0 m/s downward
  + Correct direction: rocket must be within a 10 degree arc of vertical, that is, the rocket must be oriented straight up and down, or can be up to 5 degrees left or right of vertical
* After each attempt, the rocket’s starting horizontal position and orientation is changed
  + This adds an element of surprise to each attempt
  + Players must quickly react to land successfully
* The level of the game cannot be changed while successful or failed landing animations are playing.
* The game can be paused

Landing Site

* Dimensions of the landing site depend on the level of the game
* The landing site controls the location range the rocket has to land

Rocket

* Rocket is given a starting downward velocity (already falling before it comes into view)
* Rocket will fall faster and faster unless thrust of main engine counteracts gravity
* Players control the rocket by engaging thrusters and the rocket’s main engine
  + Thrusters control the rotate of the rocket
  + The engine controls the vertical and horizontal thrust
* Players control the level of thrust the main engine produces
  + Higher thrust more quickly overcomes force of gravity
  + Thrust level has a maximum value and a minimum value
  + The minimum value is zero thrust
* Rocket requires rocket fuel to use main engine and monopropellant to use thrusters
  + If a fuel type runs out, control is lost for the fuel’s respective equipment
* Fuel levels decrease based on use
  + Rocket fuel decreases proportional to the thrust level (higher thrust level means more fuel is used when the engine is engaged)
  + Monopropellant decreases at a constant rate

**Controls:**

The gameplay can be controlled with the following input:

* SPACEBAR: pauses or unpauses the game

The rocket can be controlled with the following input:

* W: engages the rocket’s main engine
* A: engages the left thruster of the rocket
* D: engages the right thruster of the rocket
* Up Arrow: increases the thrust level
* Down Arrow: decreases the thrust level

**Platform:**

Rocket Lander has the following expected equipment:

* Computer with keyboard
* Internet access

**How Our Game Is Different or Unique**

The concept of landing a rocket is quite novel. Very few 2D games have been created that focus on this. We know of two games that have the same theme of landing a rocket.

The first also simulates the SpaceX landing concept of landing in the ocean.

* + Their game was built using 8-bit graphics, so the look and feel of our game is very different. We use a much higher resolution rocket and set of images, as well as more robust animations. We show the landing legs unfolding, which adds a sense of realism to gameplay. Our rocket thrust animation varies depending on thrust level.
  + Their game was designed so the rocket is very small and has a large distance to fly to the landing site. Our game focuses on a much smaller range of view, resulting in a larger rocket, which we believe will be more engaging for players. Our game focuses on the most important part, the crucial moments before landing.
  + Their game was designed with more realistic physics, including the main engine’s ability to gimbal (affect rotation) and the rocket’s inability to recover after exceeding rotation limits. However, this results in a frustratingly difficult situation regarding control of the rocket. Our game simplies the physics and gives players greater ability to control the orientation of the rocket. Though unrealistic, it makes control feedback much more obvious and engaging for a wider audience age.
  + Their game does not vary the amount of thrust coming from the main engine, making it very difficult to successfully land and precisely control the rocket.
  + Their game also has a much more narrow range of successful landing conditions, again, making it difficult to complete a landing.

The second is an outer space game where players attempt to land on a planet surface.

* + The look and feel of our games are completely different. Their game changes view range depending on how far the rocket is from the surface. We feel this is disorienting and takes away from the continuity of movement. Their game uses a rocket similar to a landing module, which looks very different from our rocket. Their game doesn’t have moving animations – rocket thrust is shown as a static graphic.
  + Standard gameplay of their game is much slower. The rocket falls through less gravity, so players don’t need as quick of response for flying and landing. We feel this is less engaging and exciting than constantly being in the final moments of landing. Our game requires completely focus, and this absorbs players into the game. It also feels like a much greater reward when the rocket is successfully landed.

Simulating semi-realistic physics in a 2D game is also unusual. Most current 2D space and rocket games focus on shooting enemies or asteroids, not about acceleration and momentum. With the focus on more realistic physics, control of the rocket becomes much more difficult. This is really the focus of our game. Our game provides a simulation of how difficult it is to control and a rocket, something many people – even space enthusiasts – don’t really think about. At the same time, it simplifies the physics so that the game is still enjoyable and simple. We want to engage people and inspire them to think about the space industry, not turn them off because the game is too difficult.

**Assets:**

Our game includes the following assets we have been learning about:

* Graphics (Bitmap / Shape)
  + Background of the level is a Bitmap image
  + Landing site is a hidden Shape
  + The visualization of Fuel Remaining are also Shapes
  + Hidden Shapes were used with the rocket to locate and store smoke generation points
* Animation
  + Fuel Remaining elements are a fill bar that changes as fuel is consumed
* Input
  + Players use two sets of controls on the keyboard to control the game and rocket
* Containers
  + The rocket was developed a container. It holds multiple sprites and Shapes and rotates and moves them altogether
* Sprites
  + The rocket body is a sprite. This sprite has animation for the grid fins extending
  + The landing legs is a separate sprite. This sprite is placed over the top of the rocket body so that the landing legs can be animated separately
  + The thruster jets are sprites.
  + The rocket engine fire is a sprite.
  + Smoke puffs are sprites.
* Sprite Animation
  + Rocket body has animation to show grid fins extending
  + Landing legs have animation to show unfurling
  + Thruster jet shows gas coming out
  + Rocket engine fire shows fire coming out, at differing sizes depending on thrust level
  + Smoke is released whenver a thruster or engine is engaged. The smoke rises and fades away for realism.

**Project Backlog**

A: animation

C: Code

G: graphics

GAME: game

GUI: GUI

M: movement

O: objects

PHYS: Physics

| Done? | Item Description | Category | Phase | Responsible |
| --- | --- | --- | --- | --- |
| No | Design high level units | CODE | 0 | Ryan |
| No | Design low level units  Goal: each unit can be readily understood (and implemented)without having to know details about other units. | CODE | 0 | Jack |
| No | Develop diagrams to illustrate relationship between program units | CODE | 0 | Ryan |
| Yes | Build spritesheet for rocket graphic | A | 1 | Ryan |
| Yes | Design the rocket body graphics with grid fins | G | 1 | Ryan |
| Yes | Design the rocket landing legs graphics | G | 1 | Ryan |
| Yes | Add ability to pause the game | GAME | 1 | Ryan |
| Yes | Develop method for resetting game after landing attempt | GAME | 1 | Ryan |
| Yes | Add controls for rocket to rotate side to side | M | 1 | Ryan |
| Yes | Add controls to engage engine | M | 1 | Ryan |
| Yes | Develop collision detection for rocket (end game checks) | M | 1 | Ryan |
| Yes | Develop movement ability of rocket (vertical, horizontal, rotational movement) | M | 1 | Ryan |
| Yes | Build Center of mass object | O | 1 | Ryan |
| Yes | Build landing site object | O | 1 | Ryan |
| Yes | Build rocket container   * Create sprite for rocket body * Create sprite for landing legs * Create sprites for left and right thrusters * Create sprite for main engine fire * Create shapes for smoke generation points | O | 1 | Ryan |
| Yes | Add vertical and horizontal velocity concepts | PHYS | 1 | Ryan |
| Yes | Develop simulation of acceleration due to gravity | PHYS | 1 | Ryan |
| Yes | Add animation method for thrusters   * Ability to flare both when landed * Ability to flare one or both due to player control | A | 2 | Ryan |
| Yes | Add animation methods for main engine   * Engine running at each thrust level * Engine cutting out after thrusting and fire animation diminishing gradually | A | 2 | Ryan |
| No | Build Explosion animations | A | 2 | Jack |
| No | Build spritesheet for explosion | A | 2 | Jack |
| Yes | Build spritesheet for fire graphic | A | 2 | Ryan |
| Yes | Build spritesheet for thruster graphic | A | 2 | Ryan |
| Yes | Change thrust animation based on thrust level | A | 2 | Ryan |
| Yes | Design engine flame graphic in multiple sizes | G | 2 | Ryan |
| No | Design explosion graphics   * One explosion for land or surface * One explosion for water? | G | 2 | Jack |
| Yes | Design the land-based background | G | 2 | Ryan |
| Yes | Design the thruster graphics | G | 2 | Ryan |
| Yes | Develop method to simulate solid surface at landing level (slice strategy) | G | 2 | Ryan |
| Yes | Develop ending sequence for failed landing | GAME | 2 | Ryan |
| Yes | Develop ending sequence for successful landing | GAME | 2 | Ryan |
| Yes | Develop methods for placing rocket in random position and orientation | GAME | 2 | Ryan |
| Yes | Build generic method for text objects | GUI | 2 | Ryan |
| Yes | Build the paused text object | GUI | 2 | Ryan |
| Yes | Build the physics statistics text object | GUI | 2 | Ryan |
| Yes | Build update methods for GUI text on screen | GUI | 2 | Ryan |
| Yes | Add controls to change thrust level | M | 2 | Ryan |
| Yes | Add thrust level concept | M | 2 | Ryan |
| Yes | Develop ability to change level of thrust | M | 2 | Ryan |
| Yes | Build land-based bitmap object | O | 2 | Ryan |
| Yes | Add altitude concept | PHYS | 2 | Ryan |
| Yes | Build spritesheet for smoke graphic | A | 3 | Ryan |
| Yes | Design animations for smoke wisp (fade and float upward), add location randomness to improve generation realism | A | 3 | Ryan |
| No | Improve rocket landing ending sequence   * Rotate rocket to be positioned straight vertically if rocket landed at an angle (Tween?) | A | 3 | Jack |
| No | Simplify engine fire animation | A | 3 | Jack |
| No | Fix bug with multiple spawning when level is reset after ending sequence | CODE | 3 | Jack |
| Yes | Organize code into logical groupings | CODE | 3 | Ryan |
| No | Reduce coupling in the code   * Reduce dependency of methods on variables, global variables | CODE | 3 | Jack |
| Yes | Add decals / logo to rocket | G | 3 | Ryan |
| Yes | Create loading screen graphic | G | 3 | Ryan |
| Yes | Design the ocean-based background | G | 3 | Ryan |
| Yes | Design the smoke wisps | G | 3 | Ryan |
| No | Add point scoring system to rate how well player landed   * Proximity to landing circle (closer is better) * Amount of fuel remaining (more is better) * Angle of rocket (more vertical is better)? | GAME | 3 | Jack |
| No | Create loading screen | GAME | 3 | Jack |
| Yes | Develop ability to change levels | GAME | 3 | Ryan |
| No | Simplify reset of game so rocket can be reused instead of rebuilt each time | GAME | 3 | Jack |
| No | Build the explanation of gameplay shown at start and when game is paused (controls and objective) | GUI | 3 | Jack |
| Yes | Build the fuel remaining text object | GUI | 3 | Ryan |
| Yes | Build the Game hint text object | GUI | 3 | Ryan |
| Yes | Build update methods for Fuel Remaining visualizations | GUI | 3 | Ryan |
| No | Develop ability to turn on and off center of mass visualization of rocket | GUI | 3 | Jack |
| No | Zoom screen to 50% when start | GUI | 3 | Jack |
| Yes | Add concept of fuel levels, decreasing fuel when using equipment, no controls once fuel rules out | M | 3 | Ryan |
| Yes | Build ocean-based bitmap object | O | 3 | Ryan |
| Yes | Build smoke sprite method for generating smoke visualization | O | 3 | Ryan |
| Yes | Design the fuel remaining visualization | O | 3 | Ryan |
| No | Improve physics simulation:   * Rocket continues to rotate after thruster stops * Rocket has horizontal momentum | PHYS | 3 | Jack |
|  |  |  |  |  |
|  |  |  |  |  |