

Final Report

Ryan Nelson

Jack Morgan

Revised: 30 April 2017

**Table of Contents**

1. **Story / Narrative**
2. **Scene**
3. **Game Objects**
4. **Game Physics**
5. **Game Events**
6. **Rules and Objectives**
7. **User Controls**
8. **Platform & Required Equipment**
9. **How Our Game is Different or Unique**
10. **Required Assets**
11. **Differences from the First project**
12. **Development Highlights**
13. **Project Features / Backlog / Responsibilities**

**1. 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 on Thursday March 30, relaunched (and relanded) one of its rockets it had landed last year.

This game is a tribute to the incredible achievements 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 the game we developed in Project 1 of this university course.

**2. Scene**

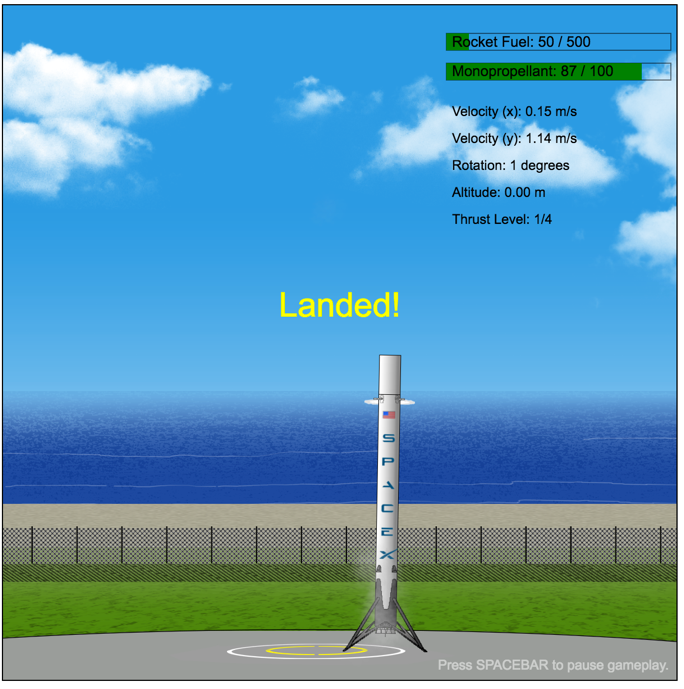


Figure : Flaring thrusters after successful landing

The game is laid out in a single frame, with all content visible. The game background will be themed based on the level, and provide a visualization of where the rocket should be landed. Players will control the rocket with the keyboard, and game is reset after each landing attempt.

The game’s GUI will include the following visualizations:

* remaining rocket fuel (main engine)
* remaining monopropellant (thrusters)
* physics statistics (horizontal and vertical velocity, current rotation, thrust level, altitude)

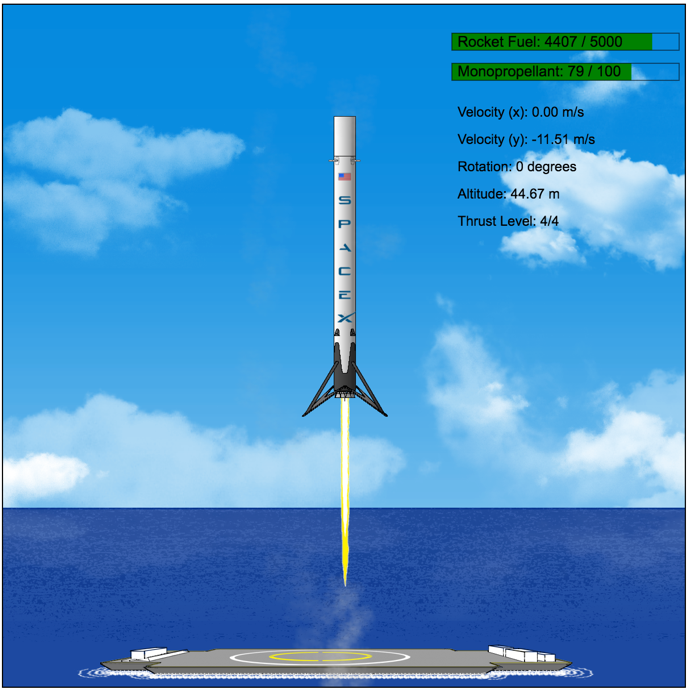


Figure : Rocket is attempting to land

**3. Game Objects**

Stage: CreateJS library uses a Stage object as a container to hold all other game objects.

Game Manager: object contains the functionality used to direct game flow

Input Manager: object contains the constants / functions related to triggering game events from user input

GUI Manager: a container to hold all GUI elements, as well as the explosion animation sprite.

Background Manager: object contains the functionality used to create and switch game backgrounds.

Collider: object contains functionality used to track altitudes of rocket and landing site, as well as detect when a landing or crash has occurred.

Smoke: instances of this object are created every time an engine or thruster fires.

Landing Site: object sets the location and altitude that a rocket can be landed.

* The width of the landing site varies depending on the current game level
* For the land-based landing site, the rocket can land at any horizontal coordinate at zero altitude.
* For the ocean-based landing site, the rocket must land on the deck of the drone ship and not on its equipment or facilities or in the ocean.

Rocket: object that players control

* Modeled directly on SpaceX 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 the engine level to vary the amount of thrust the rocket engine releases.
  + High thrust uses more rocket fuel than low thrust does.
  + High thrust contributes a greater force to the rocket, which results in a faster change of direction.
* Players control the firing of the engine to generate thrust and counteract the effects of gravity and momentum.
  + If the rocket runs out of rocket fuel, the engine shuts off and no longer fires.
* Players control two cold-gas thrusters at the top of rocket to rotate the rocket and redirect the thrust of the engine.
  + If the rocket runs out of monopropellant, the thrusters shut off and no longer fire.
* Rocket has animated elements, including grid fins, landing legs, thrusters, and engine flames.

SpriteSheets: these objects contain information and images needed for animation

**4. Game Physics**

The game simulates the following physics phenomena:

* Gravity: The rocket falls to the surface at a continuously increasing rate due to this force.
* Momentum:
  + Horizontal: If engine thrust is stopped, the rocket continues to move horizontally in the direction it was traveling. It takes a certain amount of thrust to counteract this momentum and reverse the horizontal direction of movement.
  + Vertical: It takes a certain amount of thrust to counteract the force of gravity and slow down the rocket. (The rocket will not immediately change directions when its engine is engaged, it must reduce downward velocity, reach zero, before it can finally can fly upward.)
* Torque: firing the thrusters at the top of the rocket induces torque on the rocket, which causes the rocket body to rotate.
* Thrust: firing the rocket engine generates thrust. The amount of thrust depends on the engine level; a higher engine level results in greater thrust. Depending on the angle of thrusting, horizontal and vertical forces are induced on the rocket, which change the rocket’s velocity vector.
* Collision: In order to land successfully, the rocket must collide with the ground under very particular conditions. Failure to do so will result in the destruction of the rocket, because a solid body has collided with another solid body.

**5. Game Events**:

The game has the following events:

* HTML onload
  + HTML body has finished loading
* Loading Screen
  + Occurs after onload
  + Loading Screen is shown
* LoadQueue is completed
  + Occurs during Loading Screen event
  + All HTMLImageElements are loaded into LoadQueue
* Game is loaded
  + Occurs after LoadQueue is complete
  + All game objects are created and positioned
  + Within a range, the rocket spawns in a random horizontal position and angle
* Gameplay explanation
  + Occurs after game is loaded
* Game start
  + Occurs after gameplay explanation
  + Starts the Ticker
  + Shows the landing leg and grid fin extension animations on the rocket
* Game Step (Tick)
  + Occurs after Ticker is started
  + Continuously performs update and rendering of game objects until game over
* Game Paused
  + Occurs when player presses spacebar
  + Pauses Ticker, GUI indicates game is paused
* Game Reset
  + Rocket is repositioned for another attempt, all text and values are reset

**5. Game Events (continued)**

* Level is changed
  + Occurs when player presses left arrow
  + Can only occur between collision events
  + Background and landing site are changed, game is reset
* Key Down (W, A, D, Up, Down, Spacebar)
  + Occurs when player presses a key
  + Causes key specific events to occur
* Key Up (W,A,D)
  + Occurs when player releases key
  + Causes key specific events to occur
* Engine level is increased
  + Occurs when Up key is pressed
  + If engine level is below 4, adds 1 to engine level
* Engine level is decreased
  + Occurs when Down key is pressed
  + If engine level is above 0, reduces engine level by 1
* Engine level is changed
  + Occurs if engine level is increased or decreased
  + If engine is firing, fire animation is updated to correspond to revised engine level
* Engine firing
  + Occurs when W key is pressed
  + If engine level is greater than 0 and there is fuel remaining, shows fire and smoke animations and adds thrust to the rocket
* Engine cutoff
  + Occurs when W key is released, when rocket runs out of fuel, or when rocket lands
  + Shows fire cutout animation, then shows no fire
  + Removes thrust from rocket
* Left Thruster Firing
  + Occurs when A key is pressed
  + If there is monopropellant remaining, shows cold-gas and smoke animations and adds positive torque to the rocket
* Right Thruster Firing
  + Occurs when D key is pressed
  + If there is monopropellant remaining, shows cold-gas and smoke animations and adds negative torque to the rocket
* Left Thruster cutoff
  + Occurs when A key is released
  + Shows cold-gas cutout animation, then no animation
  + Removes positive torque from rocket
* Right Thruster cutoff
  + Occurs when D key is released
  + Shows cold-gas cutout animation, then no animation
  + Removes negative torque from rocket

**5. Game Events (continued)**

* Rocket Update
  + Occurs at each game step event
  + Rocket’s next angle is determined based on torque forces
  + Rocket’s next position is determined based on horizontal and vertical forces
  + Without vertical forces added by rocket engine, rocket accelerates downward
  + Values are stored
* Rocket Render
  + Occurs at each game step event
  + Rocket’s next angle and position values are used to change the actual angle and position of the rocket
* Rocket has landed
  + Occurs when rocket has collided with the landingSite and meets specific conditions of velocity, angle, and position
  + Stops rocket movement, removes velocity vectors, then flares thrusters
  + Game is reset automatically after a period of time
* Rocket has crashed
  + Occurs when rocket reaches zero altitude and fails to meet landing conditions or misses the landingSite
  + Explosion animation is shown
  + Game is reset automatically after a period of time
* Smoke has been added
  + Smoke object drifts upward and fades out
  + After fading out, smoke object is removed

**6. Rules and Objectives**

Game

* The goal of the game is to land the rocket first stage on the landing site. The following criteria for a “successful” landing are used:
  + Correct vertical position: one or more of landing legs of rocket have 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 vertical velocity: rocket cannot be traveling faster than 2.0 m/s downward
  + Correct horizontal velocity: rocket cannot be traveling faster than 2.0 m/s horizontally
  + Correct orientation: 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
* Game is reset after each success or failed landing attempt
  + rocket is repositioned in a random horizontal position and orientation, the rocket’s vertical starting position remains unchanged for each attempt
  + This adds an element of surprise to each attempt
  + Players must quickly react to land successfully
* The level of the game cannot be changed during Rocket Landed or Rocket Crashed events.
* The game can be paused. This stops all rocket movement and animation.

**6. Rules and Objectives (continued)**

Landing Site

* Dimensions of the landing site depend on the level of the game
* The landing site controls the coordinate range the rocket has to land
* The landing site object is not visible during gameplay; background images represent landing site

Rocket

* Rocket is given a starting downward velocity (already falling before it comes into view)
* Rocket will fall faster and faster due to gravity
* Main engine can be fired to add thrust force which can counteract gravity and change the position of the rocket
* Left and Right Thrusters can be fired to add torque to rocket, which rotate the rocket side to side
* Players control the rocket using thrusters and the rocket’s main engine
* 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

**7. Game Controls:**

The gameplay can be controlled with the following input:

* SPACEBAR: pauses or unpauses the game
* Left Arrow: changes game level

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 engine level
* Down Arrow: decreases the engine level

**8. Platform & Required Equipment:**

Rocket Lander has the following expected equipment:

* Computer with keyboard
* Internet access
* A browser with a window that can be zoomed out to 50%

**9. 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 similar game 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.
  + 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 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 slightly 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 similar game 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 gaming experience is stressful.

**10. Required Assets:**

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

* Graphics (Bitmap / Shape)
  + Background of the level is a Bitmap image
  + Smoke puffs are bitmap images
  + Landing site is a hidden Shape
  + The visualization of Fuel Remaining includes 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
  + Smoke is released whenever a thruster or engine is engaged. The smoke rises and fades away for realism.
* Input
  + Players use two sets of keys on the keyboard to control the game and rocket
* Containers
  + The rocket was developed as a container. It holds multiple Sprites and Shapes and rotates and moves them altogether
  + The GUI will also be developed as a container, holding all Text objects and Fuel Remaining elements
* 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 individual sprites.
  + The rocket engine fire is a sprite.
* 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

**11. Differences from the First Project**

1. The largest difference is that the code in Project 2 is much cleaner than Project 1. Encapsulating behavior inside IIFEs and objects made the code much more readable and easier to navigate and debug.
2. Project 2 has better game flow. If one compared the way the Loading Screen from Project 1 transitioned into the actual game against the way it works in Project 2, the second version is much more straightforward.
3. Project 2 is also more efficient with game changes.

Project 1 involved deleting all game objects every time and rebuilding them. In Project 2, we simply reposition the rocket and reset its values.

In Project 2, we focused on developing a design where objects were instantiated once and only object visibility is changed when objects need to be shown or hidden.

* For the Shape object making up the FuelBars, only the scale of that object is changed, rather than the object being redrawn every time the fuel level changed.
* Game background images are loaded once, and then only their visibility is changed to switch between levels

**12. Development Highlights**

1. Background Images and Loading Screen
   1. These were illustrated custom images for the game and add a high level of finish. We put a lot of effort into creating beautiful images.
2. Rocket animations
   1. We illustrated custom images for all frames of the Rocket sprite animations (see image on following page). These were drawn using AutoCAD and Photoshop and were based on the exact dimensions of SpaceX Falcon 9 rockets.
   2. Animations (flames moving, legs extending, fins extending and collapsing) had to be drawn without tracing.
   3. We developed a sophisticated animation machine which would transition between multiple flame sizes and situations (see Design Document Diagram 18) for the rocket engine.
3. Smoke generation feature
   1. We developed a system that whenever an engine or thruster would fire, realistic looking smoke would be generated.
   2. To do this, we added hidden Shapes to the Rocket container which located these end points. This solved the project of being able to calculate where these points were regardless of rocket rotation or movement. Inside the given firing function, we make a call to generate a smoke bitmap and pass the hidden Shape object that corresponded to the firing.

Figure : End points

* 1. A semi-transparent Bitmap is generated at a random location near that point (for realism), then the Bitmap rises and fades using a Tween. Finally, the Tween calls a function which deletes the Bitmap from the game.

1. Design Diagrams
   1. We spent a lot of time working out the clearest way to organize and represent the game and its pieces.

Diagram showing all of the different combinations of sprite frames that were developed for the Rocket.

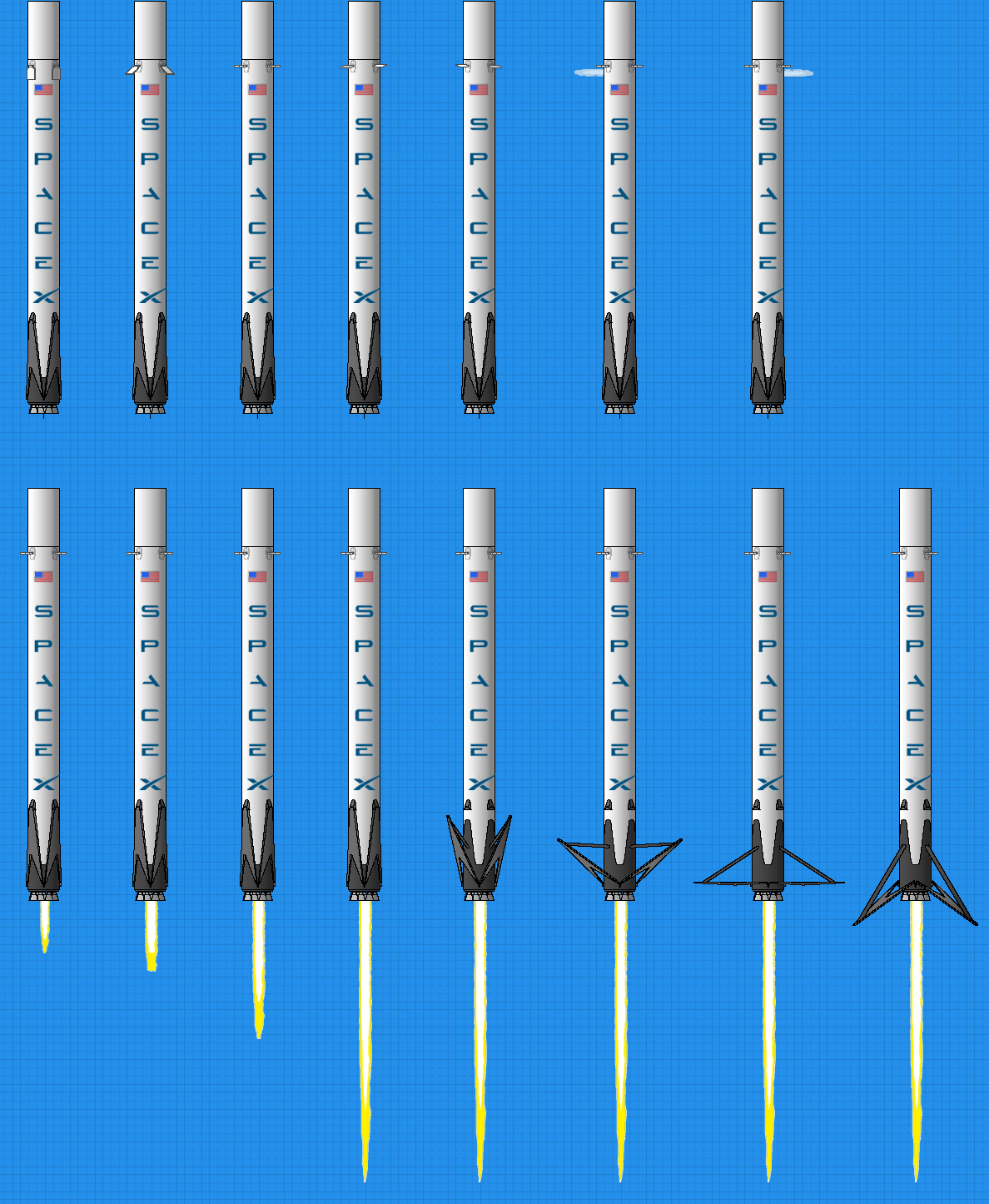
* Grid fins (near the top of the rocket) can be extended or retracted. Grid fins tilt when rocket rotates.
* Landing legs go through a series of extending frames.
* Two cold gas thrusters are animated independently.
* Rocket engine has variable levels and has a flame to match each engine level.

Figure : Custom animation frames for Rocket

1. **Project Features / Backlog / Responsibilities**

| Done | Item Description | Category | Step | Responsible |
| --- | --- | --- | --- | --- |
| Yes | Build first version of the finished game | \_CONCEPT | 0 | Ryan |
| Yes | Method to simulate solid surface at landing level (slice strategy) | \_CONCEPT | 0 | Ryan |
| Yes | Specification Doc | \_DESIGN | 1 | Ryan |
| Yes | Class diagram | \_DESIGN | 2 | Ryan |
| Yes | High level Design diagram | \_DESIGN | 2 | Ryan |
| Yes | Game Object diagram | \_DESIGN | 2 | Jack |
| Yes | High Level Game Flow diagram (Events, Functions?) | \_DESIGN | 2 | Jack |
| Yes | Revise game flow diagram | \_DESIGN | 2 | Ryan |
| Yes | Revise game object diagram | \_DESIGN | 2 | Ryan |
| Yes | Update object diagram | \_DESIGN | 4 | Ryan |
| Yes | Update game flow diagram | \_DESIGN | 4 | Ryan |
| Yes | Update specification document for final submission | \_DESIGN | 5 | Ryan |
| Yes | Update object diagram for final submission | \_DESIGN | 5 | Ryan |
| Yes | Update game flow diagram for final submission | \_DESIGN | 5 | Ryan |
| Yes | Build spritesheet for rocket graphic | ANIMATION | 3 | Ryan |
| Yes | Build spritesheet for fire | ANIMATION | 3 | Ryan |
| Yes | Build spritesheet for thruster | ANIMATION | 3 | Ryan |
| Yes | Design animations for smoke wisp (fade and float upward), add location randomness to improve generation realism | ANIMATION | 3 | Ryan |
| Yes | Build spritesheet for explosion | ANIMATION | 5 | Jack |
| Yes | Develop collision detection object   * All functions, properties to do with collision * Trigger events if collision occurs | COLLIDER | 3 | Ryan |
| Yes | Track rocket position relative to landing site (altitude) | COLLIDER | 4 | Ryan |
| Yes | Build FuelBar IIFE   * Shapes / functions to visualize fuel levels | FUEL BAR | 4 | Ryan |
| Yes | Build Game Manager IIFE to contain all properties, constants, variables, and functions related to gameplay   * Running game * Resetting game * Pausing game * Ending sequences * Move all current game functions into this object | GAME | 4 | Jack |
| Yes | Develop ability to change levels | GAME | 4 | Jack |
| Yes | Develop Game Manager IIFE (additional methods and features) | GAME | 4 | Jack |
|  | 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) * Velocity at landing (smaller is better) | GAME | 6?  Nice  To  Have | Jack |
| Yes | Zoom screen to 50% when open game in browser | GUI | 3 | Ryan |
| Yes | Build GUI IIFE (Hold Text objects, Fuel Remaining visualizations) | GUI | 4 | Jack |
| Yes | Build the paused text object | GUI | 4 | Jack |
| Yes | Build the physics statistics text object | GUI | 4 | Jack |
| Yes | Build update methods for GUI text on screen | GUI | 4 | Jack |
| Yes | Build generic method for text objects | GUI | 4 | Jack |
| Yes | Build the game hint text object | GUI | 4 | Jack |
| Yes | Build the explanation of gameplay shown at start and when game is paused (controls and objective), add to game | GUI | 4 | Jack |
| Yes | Build sprite for explosion, add to GUI manager, add functions to GUI manager | GUI | 5 | Jack |
| Yes | Draw rocket graphics (Body, Grid fins, Landing Legs, SpaceX Logo) | IMAGE | 2 | Ryan |
| Yes | Draw thruster graphics | IMAGE | 2 | Ryan |
| Yes | Draw fire graphics (Multiple sizes correspond with engine level) | IMAGE | 2 | Ryan |
| Yes | Draw the land-based background graphic | IMAGE | 2 | Ryan |
| Yes | Draw smoke graphic | IMAGE | 2 | Ryan |
| Yes | Draw loading screen graphic | IMAGE | 2 | Ryan |
| Yes | Draw the ocean-based background graphic | IMAGE | 2 | Ryan |
| Yes | Draw Explosion graphics | IMAGE | 5 | Jack |
| Yes | Develop Input IIFE   * Put all constants, variables, properties, and functions related to input into one object * Move all current input functions into this object | INPUT | 4 | Jack |
| Yes | Develop Input Manager IIFE (additional methods and features) | INPUT | 4 | Jack |
| Yes | Add input to change game level | INPUT | 4 | Jack |
| Yes | Create loading screen, add to game | LOADING | 5 | Jack |
| Yes | Build smoke bitmap method for locating, generating smoke | OBJECTS | 3 | Ryan |
| Yes | Build landing site object | OBJECTS | 3 | Ryan |
| Yes | Encapsulate smoke functions inside smoke object | OBJECTS | 4 | Ryan |
| Yes | Build land-based bitmap objects (background, slice) | OBJECTS | 4 | Ryan |
| Yes | Build ocean-based bitmap objects (background, slice) | OBJECTS | 4 | Ryan |
| Yes | Add vertical and horizontal velocity, momentum concepts | ROCKET | 3 | Ryan |
| Yes | Develop simulation of acceleration due to gravity | ROCKET | 3 | Ryan |
| Yes | Build Rocket Body Sprite | ROCKET | 3 | Ryan |
| Yes | Build Landing Legs Sprite | ROCKET | 3 | Ryan |
| Yes | Build Left and Right Thruster Sprites | ROCKET | 3 | Ryan |
| Yes | Build Fire Sprite | ROCKET | 3 | Ryan |
| Yes | Build points for smoke generation for engine fire levels | ROCKET | 3 | Ryan |
| Yes | Build points for smoke generation for thrusters | ROCKET | 3 | Ryan |
| Yes | Build Rocket container as an IIFE (objects, functions, properties, variables, constants associated with rocket) | ROCKET | 3 | Ryan |
| Yes | Build functions for rocket properties | ROCKET | 3 | Ryan |
| Yes | Build positioning and movement update methods | ROCKET | 3 | Ryan |
| Yes | Build listener addition function | ROCKET | 3 | Ryan |
| Yes | Build thruster and engine firing, cutout methods | ROCKET | 3 | Ryan |
| Yes | Build landing or crash methods | ROCKET | 3 | Ryan |
| Yes | Add animation methods for thrusters | ROCKET | 3 | Ryan |
| Yes | Add animation methods for main engine | ROCKET | 3 | Ryan |
| Yes | Change thrust animation based on thrust level | ROCKET | 3 | Ryan |
| Yes | Add thrust level concept | ROCKET | 3 | Ryan |
| Yes | Develop ability to change level of thrust | ROCKET | 3 | Ryan |
| Yes | Add concept of fuel levels, decreasing fuel when using equipment, no controls once fuel rules out | ROCKET | 3 | Ryan |