Assignment 3 - Lunar Lander

Published Edit

Introduction

This is second of two projects to help develop your game programming skills and knowledge in preparation for the final project. The purpose of this project is to expand the scope of techniques into a more sophisticated game, in preparation for the much larger scope for the final project.

You are going to create an HTML5 web-based version of a game that I used to play in my childhood on my TRS-80. The TRS-80 had limited graphics, 128 x 48 (rectangular) pixels, difficult to make graphically engaging games. While it was limited in capability, some good games were made for it, including a lunar lander style game.

Write an Lunar Lander game. The following wiki page offers a history of the concept and game: https://en.wikipedia.org/wiki/Lunar_Lander_%281979_video_game%29 (https://en.wikipedia.org/wiki/Lunar_Lander_%281979_video_game%29)

Your assignment is to write a visually modern version of the game, playable within a web browser. Your game must be visually impressive, play great, and run smoothly!

Gameplay Requirements

The goal of the game is to safely land the lunar lander. *The game starts with the lander rotated on its side, with no momentum, but gravity starting to accelerate it towards the lunar surface.* The player can rotate the lander and fire its engine. In order to safely land it must land on a flat surface, with a speed less than 2 m/s (meters per second), and an (ship) angle between 355 and 5 degrees.

- Graphics Assets
  - Must use a visually interesting background. Hubble space images are great resources!
  - Must use an interesting lander.
- Must utilize particle effects in at least two ways...
  - Thrust from the ship. Thrust must be in the appropriate direction vector and have a normally distributed spread about the direction vector (the example video demonstrates this).
  - Explosion when the ship crashes
- You can determine the acceleration of gravity and ship thrust for your game, but it should be substantially similar to the example gameplay video.
  - The biggest mistake I see on these in terms of game design is that a tiny amount of thrust sends the ship flying away too quickly, don't have the thrust acceleration be too much, let the player thrust the ship.
- Keyboard controlled
  - See technical requirements below for user control customization
- Ship Status rendering
  - Fuel : green if any left, white when empty
  - Vertical Speed : White above 2 m/s, green otherwise (green is safe landing speed)
  - Angle : White between 5 and 355, green otherwise (green is safe landing angle)
- Safe landing is when ship lands...
  - On one of the pre-defined safe landing zones
  - Speed less than 2 m/s
  - Angle between 355 and 5 degrees
- Don't worry about what happens when/if the ship moves beyond the borders of the visible canvas.
- Two levels of increasing difficulty
  - The first level must provide two safe landing areas
  - The second level has a single safe landing area that is smaller than the first safe landing areas
  - Show the user a countdown (3, 2, 1) before transitioning to the second level
  - You are free to add additional levels of difficultly, but not required
- The game ends when...
  - The ship safely lands after the second level. Upon safe landing, player can no longer control the ship; rotation and thrust controls controls disabled.
  - The ship crashes

Technical Requirements

- Menuing system that allows the player to...
  - Start a new game
  - View high scores
  - Customize controls
  - View credits (most important part of any game!)
  - Allow mouse navigation. Keyboard navigation is fine, but not required
- High scores must persist to the browser's local storage; keep at least the top 5 scores
  - You can devise a scoring system of your own
- Must use sound effects
  - Sound effect on thrust
  - (pleasing) Sound effect upon safe landing
  - Sound effect on crash
- Lunar Surface must be procedurally generated, including placement of landing zones; using the random mid-point displacement algorithm. The following is a link to the midpoint displacement algorithm presentation slides: link
- Create a particle system that manages all ongoing effects. The particle system must expose functions like, `shipThrust(...)`, `shipCrash(...)`. You may start with the particle system code I have provided, but be aware that it does not meet the requirements necessary for this assignment.
- User ability to configure the controls.
  - Required Defaults
    - Arrow left - Rotate Left
    - Arrow right - Rotate Right
    - Arrow up - Thrust
  - The interface for this must present a screen where the name of the game control is displayed (thrust, rotate left, rotate right) and to the right of it, the key used for the control. Using the mouse, the user can select the action, then some visual will change to indicate it is possible to now select a new key combination, then the user presses the new key combination (doesn't have to be a combination, could be a single key) and that immediately becomes the new keyboard shortcut for that game function.

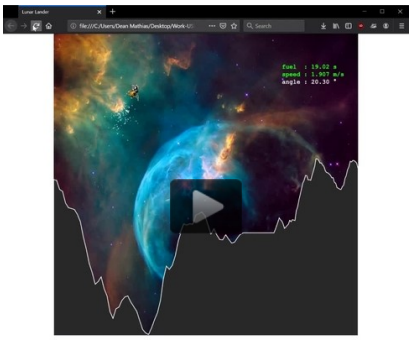
Line-Circle Intersection

The following code performs a line-circle intersection test. You are free to use this for your collision detection in the game.

```
// Reference: https://stackoverflow.com/questions/37224912/circle-line-segment-collision
function lineCircleIntersection(pt1, pt2, circle) {
    let v1 = { x: pt2.x - pt1.x, y: pt2.y - pt1.y };
    let v2 = { x: pt1.x - circle.center.x, y: pt1.y - circle.center.y };
    let b = -2 * (v1.x * v2.x + v1.y * v2.y);
    let c =  2 * (v1.x * v1.x + v1.y * v1.y);
    let d = Math.sqrt(b * b - 2 * c * (v2.x * v2.x + v2.y * v2.y - circle.radius * circle.radius));
    if (isNaN(d)) { // no intercept
        return false;
    }
    // These represent the unit distance of point one and two on the line
    let u1 = (b - d) / c;
    let u2 = (b + d) / c;
    if (u1 <= 1 && u1 >= 0) {  // If point on the line segment
        return true;
    }
    if (u2 <= 1 && u2 >= 0) {  // If point on the line segment
        return true;
    }
    return false;
}
```

Video Reference

The video below shows a substantial representation for what gameplay should be like. It isn't everything, but provides a good reference.



Development Milestones

The following are recommended development milestones to ensure you complete the project:

1. Basic menuing: by 2/22/2020
2. Terrain generation and rendering: by 2/26/2020
3. Ship rendering & control: by 2/29/2020
4. Particle system: by 3/11/2020
5. Full completion: by 3/14/2020

These are to help set milestones to achieve before it is due, they don't necessarily provide enough to suggest you'll get the game done if you only complete the milestones by those dates. Work early and often on this game.

Other General Comments

- When ESC is pressed during gameplay, the game must pause and give me the option to quit or continue. You are not required to provide this capability, but if you do, it should work like this.
- When pressing ESC in menu navigation, it should go back one level per press, not all the way back to the main menu. The probably doesn't apply for this game, menus are likely only one level deep.
  - It isn't required that ESC is supported for menu navigation, as long as there is some (obvious) way to navigate to the previous screen.
- Be sure to scale the gameplay area based on the size of the browser. Don't enforce the browser has to be at a magical size, let the user decide (within reason).
  - Can listen to window 'resize' events. Then set the canvas width/height based on the window innerWidth and innerHeight;
- Allow the mouse to be used for menu navigation, keyboard is optionally nice, but mouse navigation is required.

Points 81
Submitting a file upload
File Types zip

Due	For	Available from	Until
Mar 18	1 student	Feb 19 at 12am	Mar 18 at 11:59pm
Mar 18	Everyone else	Feb 19 at 12am	Mar 18 at 11:59pm

Assignment 3 - Lunar Lander
You've already rated students with this rubric. Any major changes could affect their assessment results.

Criteria	Ratings		Pts
Works on both Firefox and Chrome. 0 points if it only works on 1 of the browsers. Ideally it should work on all of MS Edge, Safari, Firefox, and Chrome.	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
Menuing New Game, High Scores, Customize Controls, Credits Use of mouse to select menu items	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
Technical - User control configuration Configurable controls, persisted to the browser	**15.0 pts** **Full Marks**	**0.0 pts** **No Marks**	15.0 pts
Technical - Terrain generation & rendering Must use random mid-point displacement algorithm	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
Technical - Background rendering	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
Technical - Lunar Lander rendering	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
Technical - Particle effects/system	**10.0 pts** **Full Marks**	**0.0 pts** **No Marks**	10.0 pts
Technical - High Scores persisted to browser	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
Gameplay - Ship controls	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
Gameplay - Ship status view	**5.0 pts** **Full Marks**	**0.0 pts** **No Marks**	5.0 pts
Gameplay - Two levels of difficulty	**6.0 pts** **Full Marks**	**0.0 pts** **No Marks**	6.0 pts
Gameplay - Level transition Countdown when transitioning to next level.	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
Gameplay - Safe landing Including transition to next level	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
Gameplay - Crash landing	**3.0 pts** **Full Marks**	**0.0 pts** **No Marks**	3.0 pts
			Total Points: 81.0