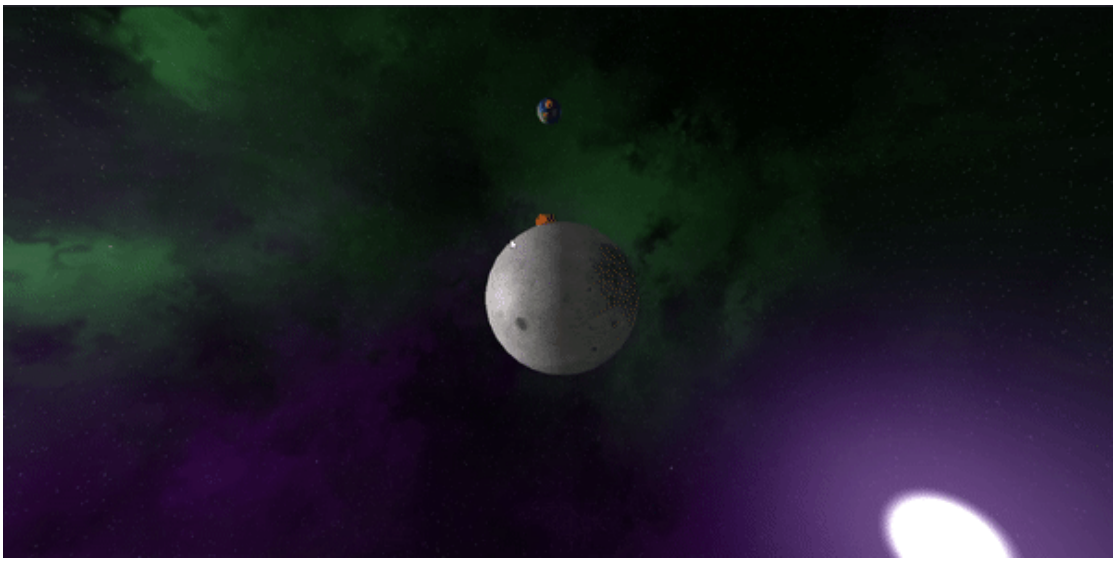
The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

# Exercise 6 – Creating with WebGL



## Overview

In this exercise you will use the elements from the previous exercise to implement a small game in WebGL!

The game will have you travel from the moon to earth, using the arrow keys, collecting stars on the way.

The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

# Modeling

You should be using your spaceship from previous exercise for your exercise. We will not grade it again - meaning if you've had point deduction for the previous exercise, we will not penalize again.

You will have to add at least 4 "star" collectibles - we suggest using shapes like dodecahedron, but you can simply use spheres (it does not matter which shape you will choose).

You will also have to model both the moon and earth - you will be required to use textures to make them look more lifelike (textures will be added).

Last but not least, keep in mind that the **hierarchy from your last exercise will not be the same necessarily in this one.**

## Gameplay Interaction

The rules for the game will be:

- The spaceship always starts at the moon and ends on earth.
- There are (at least) 3 different routes from the moon to earth, each using their own Bezier curve from the center of each planet.
- The spaceship moves along one of the bezier curves at all times.
- The spaceship may move from one route to another using the arrow keys (Left/Right)
- The spaceship needs to collect the "stars" in each route. Each star will add one point.
- When the spaceship reaches earth, there's a prompt that notifies the player how many points he accumulated.
- We recommend that the moon will be positioned at (0,0,0) and earth will be positioned at (100, 5, 100).

# Viewing

The player perspective should have a good view of all the routes so the player will be able to switch and get all the stars. The camera should be following the spaceship movement towards earth, but only on a certain axis - the camera shouldn't be affected by movement from route to route.

The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

## Materials

As we mentioned before, you will have to use **MeshPhongMaterial** for your objects. Like last time, we are giving you some artistic freedom and you may color all the objects in any color you'd like **other than the moon and earth**, which you are required to use a specific set of textures.

## Lighting

You are required to add at least two light sources:

- **Directional Light** that will represent the sun.
- **Spot Light** that is traveling on top of the spaceship.

You may add more lighting sources other than those.

## Bezier Curves

Creating the routes should be simple - use the Quadratic Bezier Curve to simply draw a bezier curve for the spaceship to travel on. Use the Curve documentations for understanding how to actually work with it.

We recommend using these control points for your routes, but you may create your curve or even use the Cubic Bezier Curve.

**Route A:** (0, 5, 40)
**Route B:** (50, 0, 50)
**Route C:** (70, -5, 70)

**Note:** You will notice when traveling between curves that the spaceship will not "feel" in the same place. This is because that the delta between each point in curve is not constant.

## How to move the spaceship

Unlike last time when we've used the animation to create an open-ended animation, this time the position of the spaceship at each frame is pre-determined according to the curves. Therefore, you will need to make sure that each frame, the movement of the spaceship will be directly to the new position on the curve.
To know where on the curve we should be, we can use the *.getPoint* method that expects a value at [0,1].
**Hint:** You have both the current world position of the spaceship (using the *position* attribute) and the world position on the curve. To create the translation matrix, some basic arithmetics may be required.

We recommend splitting all the curves to a constant number of increments between 3,000 to 10,000 with each animation frame moving a single increment.

The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

**Note:** Since we are working by frames and not by time, each computer may react differently speed-wise. You may use some coding tricks to bypass that, but it is not required (see bonus)

## Placing the collectible stars

We require **at least** 5 collectible objects ("stars") in the spaceship curves. The collectibles should be in a specific increment of a curve.

We recommend implementing it in such a way that there's a data structure that holds:

- The curve it belongs to (reference)
- The *t* value on the curve it appears at
- The Object3D of the star

And of course, a list that holds all the stars.
**Hint:** Ordered list by *t* value may prove handy later on.

## Collision Interaction

You may wonder how we can actually detect when two objects collide with each other. This usually takes some kind of physics or ray-casting techniques to resolve, but we will use a much simpler solution.

Since we know at each time what is the current *t* value and which route the spaceship is at, we can check only whenever we are expecting a collision - meaning there's a star with a similar *t* value to the spaceship. If there's a star with the same *t* value **and** route as the spaceship, we can consider it a collision and therefore we can use the *.visible* attribute to make it *false*, making it disappear.

# Bonus Features

We are adding a list of suggestions to add to your game, each has a different point value. **If you choose implementing one of the bonuses, add a README that states which bonus you've implemented.**

- Making the planets spin around themselves (1 point)
- Add more scenery (1 point)
- Make your game time-dependent instead of framerate dependent. (2 points)
- Add "bad stars" that decrease your score instead of increasing it (3 points)
- Make the spaceship go slower/faster with the ArrowUp/ArrowDown keys**.** You may have min/max speed. (3 points)

# Recommended Milestones

While it may look simple, this exercise might be overwhelming. We recommend completing it by the milestones, and not doing everything at once:

The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

1. Start with importing your spaceship from the previous exercise into the scene. You may notice you already got a skybox in this template. If you want to replace it, you can try and do it by yourself using this website.
2. Create 2 new spheres for the planets and add the textures for them.
3. Add lighting to the scene, to make sure everything is rendered properly
4. Time to add bezier curves - Start with a single curve. We recommend at first rendering the curves for visualizing - eventually you will have to remove it, but this is a good helper tool.
5. Now, create a simple animation of the spaceship traveling along the bezier curve.
6. After you have a traveling spaceship from moon to earth, time to make sure you have a good view of it - make sure that the camera follows the spaceship in both position and view.
7. Add the rest of the curves and build the traveling between curves using the keyboard. You may implement it either in a circle (after reaching to the rightmost curve return to the leftmost) or not.
   **Hint:** Create a list with all the curves, and another variable that holds the index of the current curve.
8. It's time to add the collectible stars. As always, start with a single star and add it to the scene on a specific increment of the curve
   **Hint:** You might find it easier to create a class as mentioned before, but it's not a must.
9. Test the "collection" of the single star - use the methods described in the "Collision Interaction" part.
10. Time to multiply it - create a for loop that creates stars and add them to a list.
    **Hint:** It is recommended to create the star list ordered by *t* value. This way, we can always check only for the next closest star (and if we "missed" it, move to the next one)
11. Add a counter for each collected star, and fire a prompt at the end of the curve.

## Tips

- Work smart and organized and add comments to your code. It is recommended to use the code block sections we wrote for you.
- Test all the time! The main advantage of WebGL is the ability to test fast and see the errors and bugs visually.
- Everything builds upon each other - don't move on to another feature before you are finished with another.
- Use version control. You might have a great version but then change something and it will ruin everything by mistake - better have backups.

## Grading

- Environment (Spaceship, Planets) - 10 points
- Implement Bezier Curves - 10 points
- Spaceship trailing bezier curve animation - 10 points

The Interdisciplinary Center: Computer Graphics Course 2022
Instructor: Prof. Ariel Shamir
Submission due: 19th of June 2022 (23:55)

- Crossing between curves using keys - 20 points
- Adding collectibles along the curves - 20 points
- Collision detection of spaceship and collectibles - 20 points
- Score keeping and score prompt - 10 points

## Submission

- Submission is in pairs
    - Zip file should include only a single script - hw6.js
    - **A short readme document if you decide to add more elements to your spaceship or add new animations explaining what you've implemented.**
- Zip file should be submitted to Moodle.
    - Name it:
    - <Ex##> <FirstName1> <FamilyName1> <ID1> <FirstName2> <FamilyName2> <ID2>
- Before submission be sure to check for updates on moodle
- **There will be a point reduction of up to 10 points for submitting incorrectly.**