Ece Alptekin

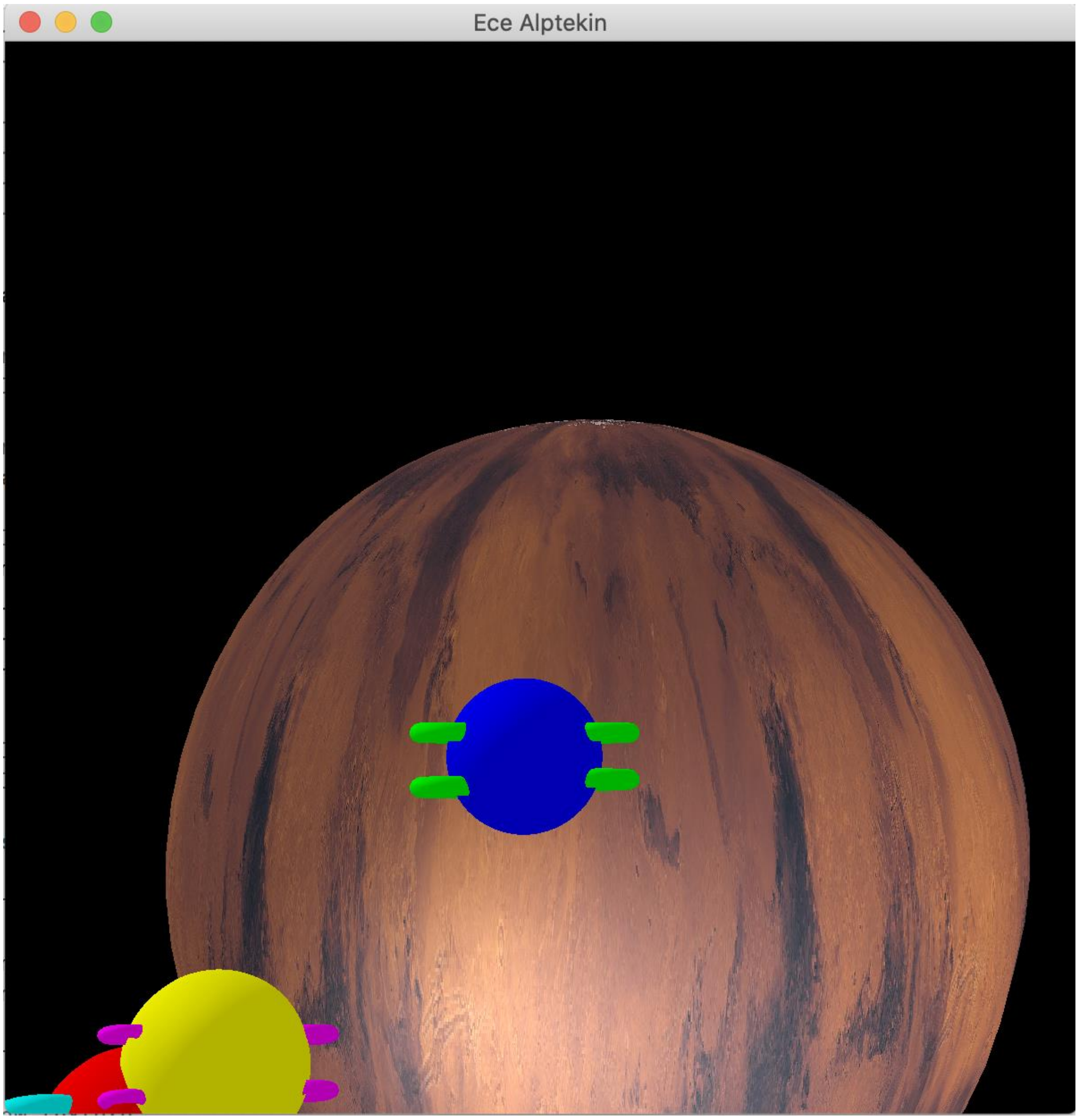
24156

3D Project Part 2

. . Sabancı .
Universitesi

Texture Mapping with Shading blending

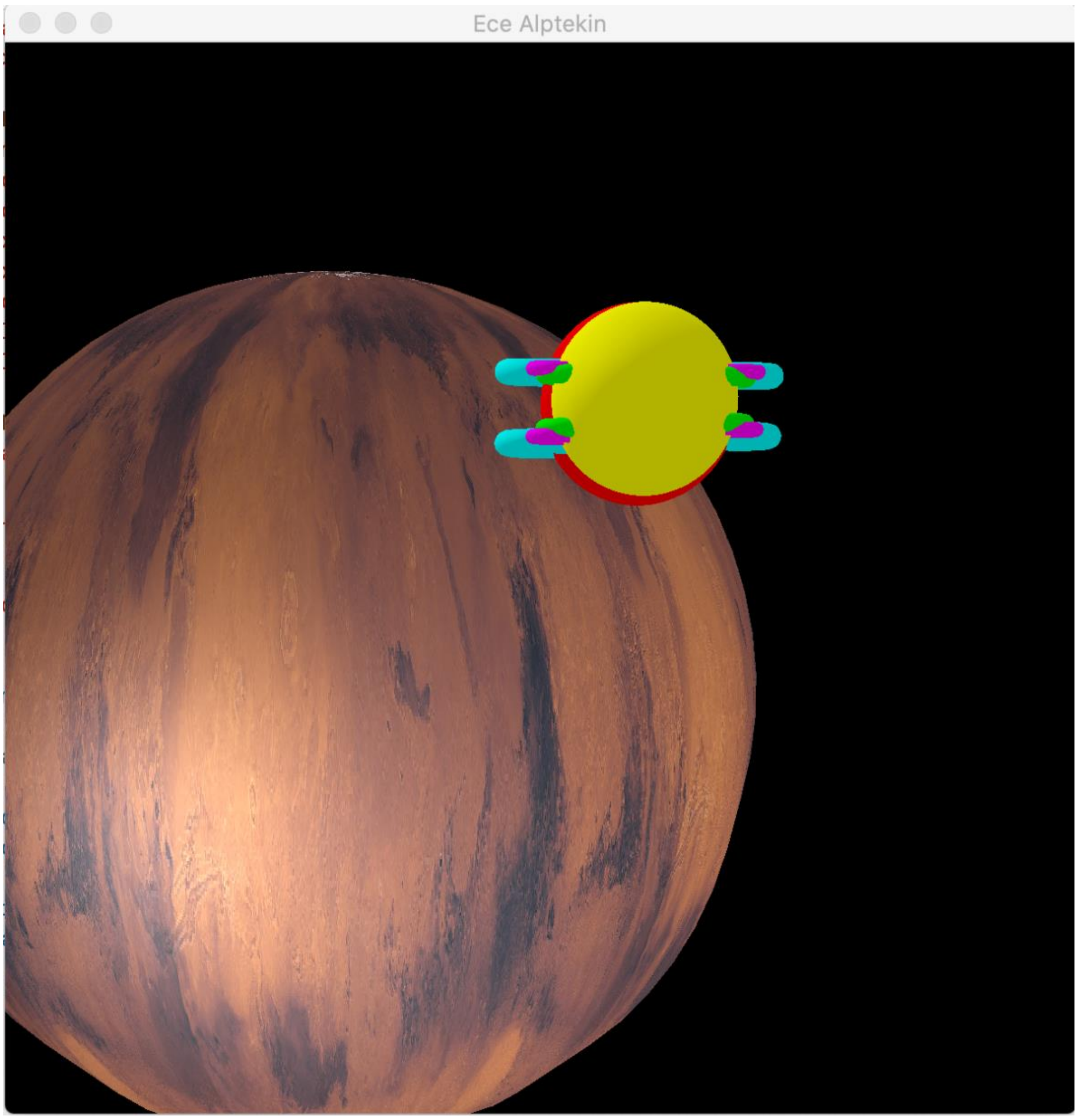
A texture as representing the surface of the Mars is added to the scene. Mars is constructed as a sphere. The texture is loaded with the functions of stb_image.header. As a filtering method, GL_NEAREST is chosen, so OpenGL selects the texel that center is closest to the texture coordinate. Texture filtering is set for magnifying and minifying operations. Blending is needed to render images with different levels of transparency. During initialization, blending is enabled and appropriate blending function are set.

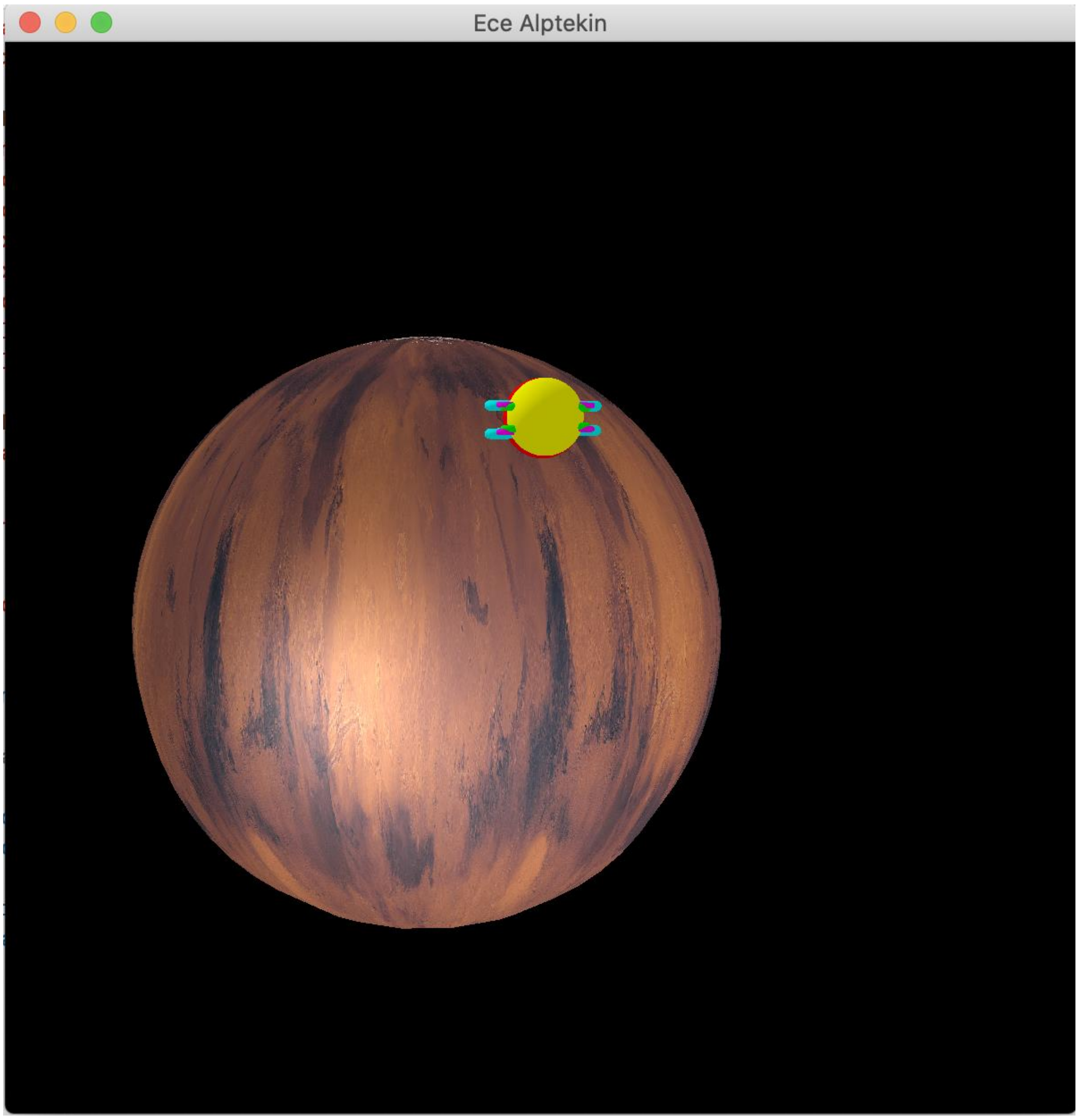


Independent Camera Control with Mouse and Keyboard

The game starts with the START mode. The user-controlled rover is defined as a child of the camera in the implementation.

To switch the camera mode, the user should firstly press key C in keyboard. After that, the user is allowed to change the camera position with the keyboards W,A,S and D. The user is also allowed to change the camera by moving the mouse. When the user presses a keyboard, the direction, the mode and deltaTime is passed to the camera header as parameters. The camera class checks the chosen direction firstly and after that the mode is controlled. If the camera mode is activated, the new position of the camera is calculated by multiplying the desired direction and velocity.

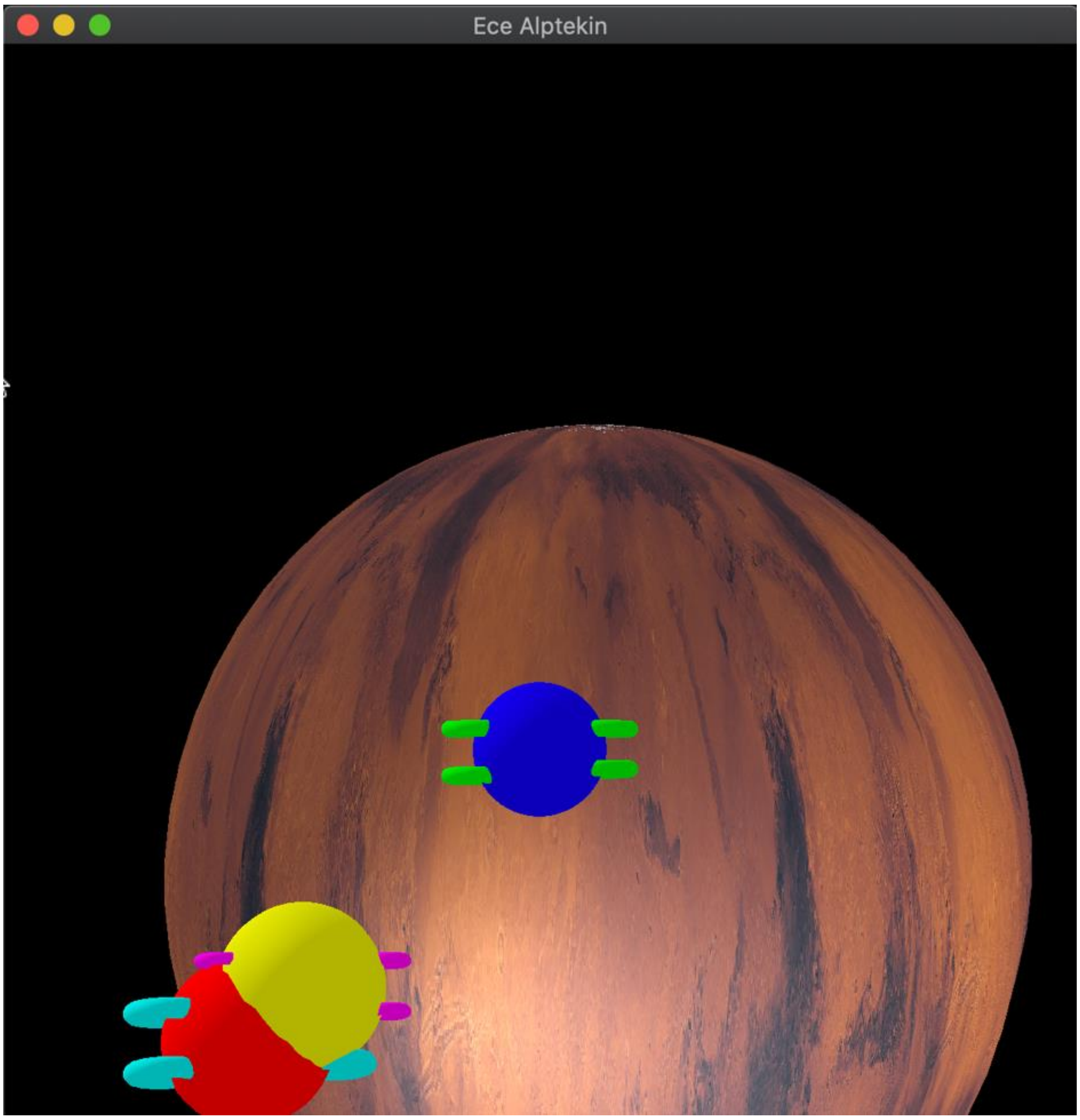




Whenever the user presses the key E, the user-controlled player stops immediately.

The user is able to change the camera position to any position by using the mouse. Additionally, the camera position is moved with a MovementSpeed to front by key W, to backward by key S, to left by key A and to right by key D.

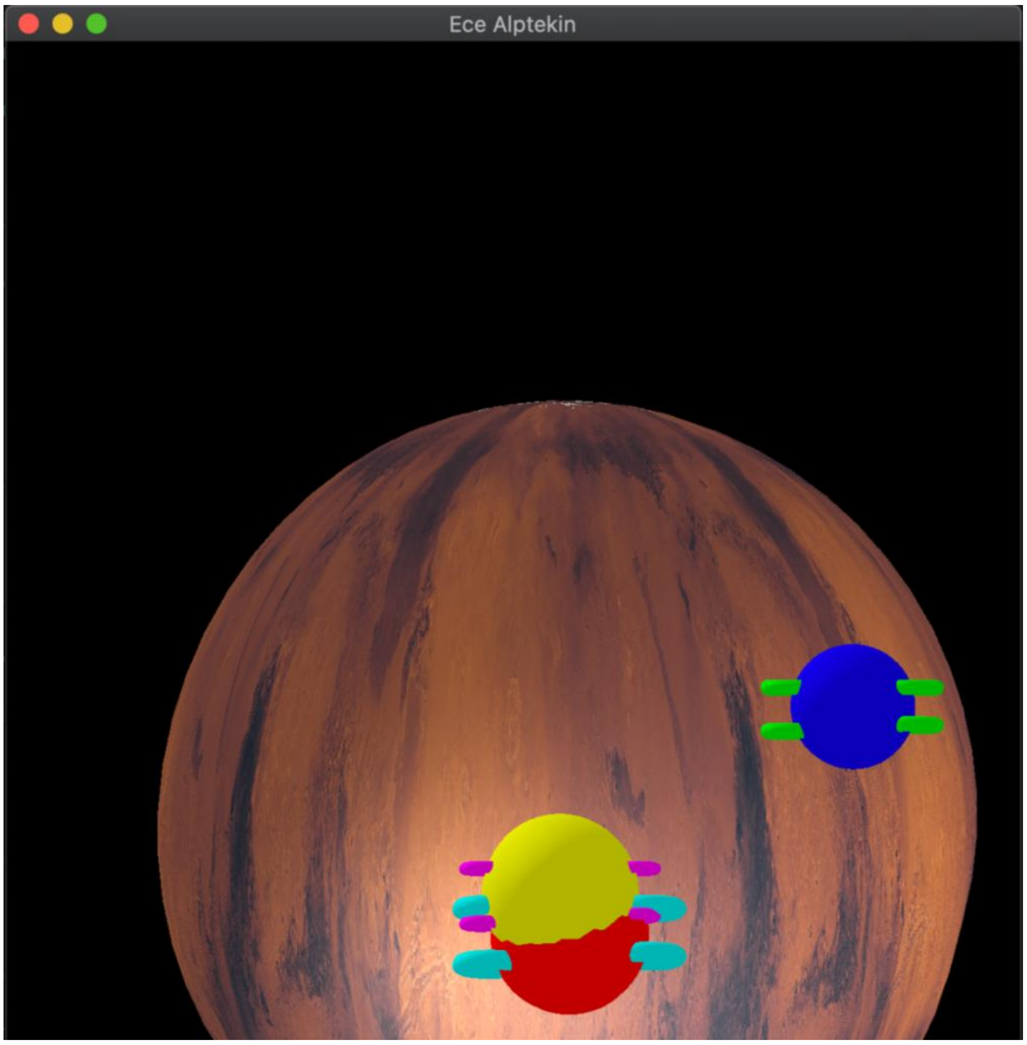
One Rover Control with Mouse and Keyboard (A Rover has 4 rotating wheels)



To switch the play mode, the user should firstly press key P in keyboard. After that, the user is allowed to change the user-controlled rover position with the keyboards W,A,S and D. The user is allowed to change the camera position by moving the mouse during playing the game. When the user presses a keyboard, the direction, the mode and deltaTime is passed to the camera header as parameters. The camera class checks the chosen direction firstly and after that the mode is controlled. If the player mode is activated, the new position of the user-controlled rover is calculated by multiplying the desired direction and velocity.

Whenever the user presses the key E, the user-controlled player stops immediately.

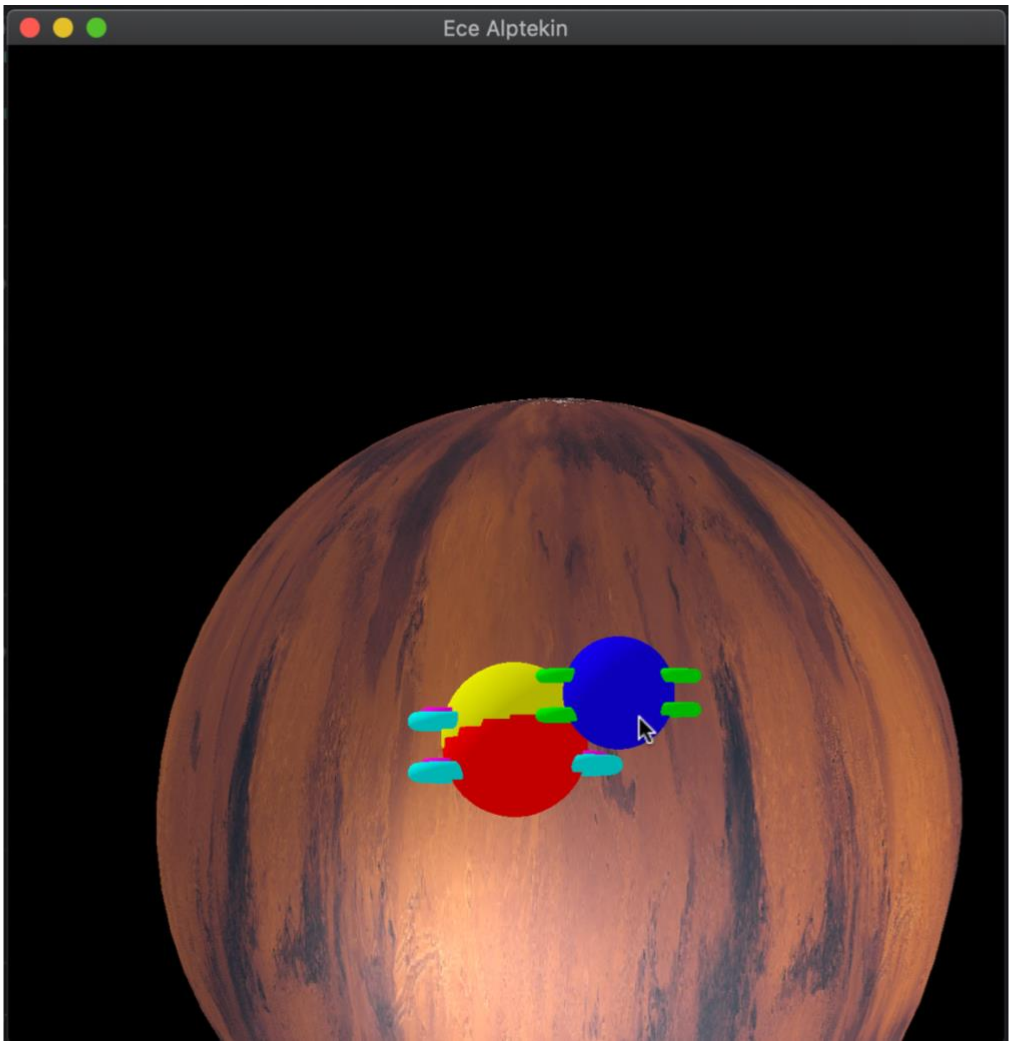
The user-controlled rover position is moved with a MovementSpeed to front by key W, to backward by key S, to left by key A and to right by key D.



Two Rovers trying to catch and collide with the user-controlled Rover, if there is a collision the user-controlled Rover stops. Use the AABB for collision detection.

Two rovers follow the user-controlled rover. When there is a collision, the user-controlled stops and is not controlled by the keyboard anymore. AABB for collision detection is used. The axis-aligned bounding boxes of two rovers are used in the implementation. The axis-aligned bounding box of each rover is calculated. When the user-controlled rover and a chasing rover enter each other's regions, the user-controlled rover stops immediately and the user cannot control it. We check if the horizontal edges overlap, and if the vertical edges overlap of both objects. If both the horizontal and vertical edges overlap, we have a collision.

After the game starts, the position of the user-controlled rover is controlled by the keyboard and the two rovers are chasing them whenever the user-controlled rover changes its position.


Ece Alptekin