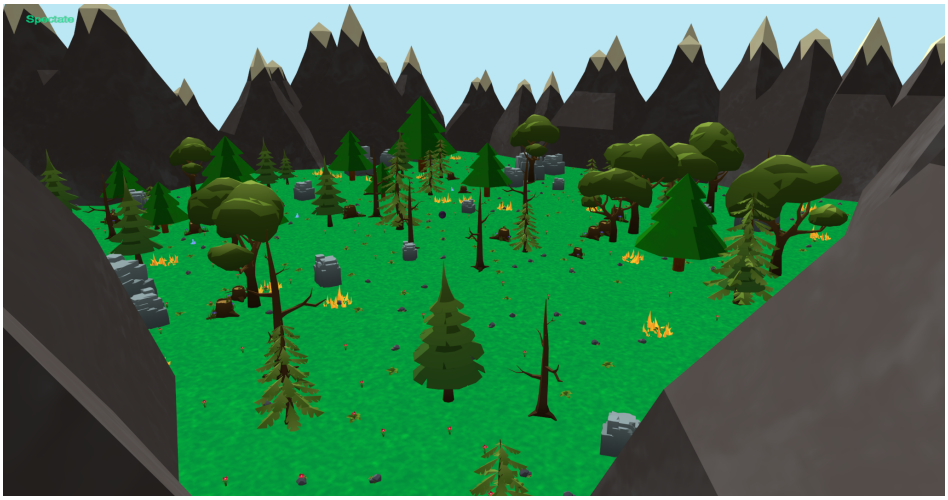

Jungle Exploration

03.07.2023

—

Sadra HeidariMoghadam

Alejandro Jaramillo

Computer Graphics Project using Vulkan

Politecnico

Overview

Jungle Exploration is a 3rd person game using Vulkan, with more concentration on beautiful cartoon graphics. The player's objective is to find the precious items and avoid the spikes to win. The game is set in a Jungle with dynamic size with random object positioning.

Goals

1. Beautiful graphics
2. Dynamic map generation
3. Smooth physics and game logics

Main C++ File

It contains the following:

1. All of the uniform buffer object structs
2. All of the vertex format structs
3. Number of each object on the map
4. Descriptor Set Layouts
5. Vertex Descriptors
6. Pipelines
7. Models
8. Textures
9. Descriptor Sets
10. Uniform Buffer Objects
11. Written texts for the main game
12. Environment Parameters containing all of the objects positions, rotations and scales.
13. Jump parameters
14. Collision parameters containing their threshold

Also the main functions for vulkan are implemented in the main file:

1. setWindowParameters:

Set the window color, size and title and aspect ratio.

2. onWindowResize:

Change aspect ratio on window resize.

3. setDescriptorPool:

Set the total number of uniform blocks, textures and sets in the pool. We use the pool for better performance, while we might have a lot of objects and textures.

a) uniformBlocksInPool = 2 + 4 + numOfPlants * 2 + numOfFlowers * 2 + numOfMountains * 2 + numOfSmallRocks * 2 +numOfStumps * 2 + numOfClouds * 2 + numOfRocks * 2 + numOfTrees * 2 * 5 + numOfItems * 2 + numOfSpikes * 2 + 4 + 1;
b) texturesInPool = 10 + 1;
c) setsInPool = 2 + 4 + numOfPlants + numOfFlowers + numOfMountains + numOfSmallRocks +numOfStumps + numOfClouds + numOfRocks + numOfTrees * 5 + numOfItems + numOfSpikes + 4 + 1;

4. localInit:

To initialize Descriptor Set Layouts bindings, Vertex Descriptors, Pipelines, Models, and textures. We can also initialize any variable we want. This function works Like the Awake function in unity engine.

5. pipelinesAndDescriptorSetsInit:

As the name suggests, in this part the pipelines and descriptor sets are created.

6. pipelinesAndDescriptorSetsCleanup:

After the game is finished, this function will be called which cleans the descriptor sets and pipelines.

7. localCleanup:

To cleanup and destroy all of the initialized textures, models, pipeline and descriptor set layouts.

8. populateCommandBuffer:

In this function, the binding between the pipelines, descriptor sets and models happens and we somehow reserve the space needed for each model to be rendered.

9. updateUniformBuffer:

   This function is called at each frame and is the main loop of the game which renders everything and calculates the logic and movements at each frame. In this part each uniform block is mapped to its descriptor set at each frame.

   The uniform blocks contain info regarding the objects world matrix and some info for the shaders input.

   This part acts like the Update method in unity engine.

Header Files

Starter:

Contains all of the vulkan library implementations. This header file makes the use of the Vulkan library easier. The complete version is available in vulkan github.

Player Handler:

Contains functions related to player movements, interactions and events, which are triggered by player interaction.

Functions are:

1. Spectate:

   the player can change the mode to spectate mode with the M key and explore the map freely.

2. Player Controller:

   In this part, all of the player movements are handled including the movement, animations and camera control. The controller moves the ball like a car with rotation animations.

3. Player Jump:

   This function is used to check if the jump is triggered by the SPACE key and is used in Player Controller.

4. Pick Item:

   This function is to check if the player is in the range of an item and if he has picked the object or not.

5. Pick Animation:

   If the object were picked, this function will be called.

6. Show Interaction Message:

   If the player was in the range of an item, a message will appear on the page.

7. Check Ending:

   Checks if all of the items were picked and shows the ending panel.

8. Check Lose:

   Checks if the player had a collision with spikes and shows the lose panel.

Collision Handler:

Contains functions to set and check collisions.

1. Map Border Collision Handler:

   Checks the collision with the border of the map and does not let the player move through the border.

2. Collision Checker:

   Checks if there is any collision while the player is moving and colliding with the objects in the map.

3. Add Collision Point:

   This function is to add collision points with each object on the map using its position and threshold(area).

Object Specifications Calculator:

Contains functions to calculate the logic behind the objects, which are not visible, including their positions, rotations, scales and if needed their collision points.

1. Calculate Environment Objects Positions And Rotations:

   To calculate the position, rotation, scales and collisions for all of the environment objects except the items on the map. These calculations are used to render the objects and pre-calculate the collisions.

2. Calculate Items Position:

   THis function is to make sure that the items will not go under another object or do not have any overlap.

3. Check Item Position Overlap:

   This function is used in the previous one to check the overlap between items and the objects.

4. Calculate Random Positions Rotations:

   To calculate random positions and rotations for each object using a random seed.

Renderer:

Contains functions to map the uniform buffer objects(ubo's) to their descriptor sets and therefore binding ubo's to the pipeline. In this part we must input the values of uniform buffer objects to render the objects in each frame.

1. Render Character:

   Calculates the rotation, position, and scale at each frame and renders the main character(the ball).

2. Render Environment:

   Calculate the environment rotation, position and scales at each frame and render all of them at each frame

   The environment contains:

   a) Ground
   b) Plants
   c) Flowers
   d) Mountains
   e) Small Rocks
   f) Stumps
   g) Clouds
   h) Rocks
   i) Trees(5 types)
   j) Items
   k) Spikes

3. Set Ubo Ds

   A function to set the variables of the uniform buffer objects with a default value.

Mesh Generator:

Contains functions to create meshes using triangle lists.

1. Create Sphere Mesh:

   To create the main character, which is a ball, using position, normal and uv coordinates.

2. Create Overlay Mesh:

   To create the 2d panels using position and uv coordinates.

Text Maker:

This is a complete header, which we used in the game to give information to the player. It gets a vertex format of 2d position and coordination and a color. It used these information to render text using a font texture.

Camera

The player camera is a 3rd person camera with 45 degrees of FOV and near plane of 0.1 and far plane of 250. The spectator camera uses a first person camera with the same info.

Shaders

Toon + Blinn(Toon)

Vertex shader

calculates the general position, fragment position, fragment norm and uv coordinates given the inputs from uniform buffers in the cpp code. Then it gives the output to the fragment shader.

Fragment Shader

Used toon BRDF model for diffuse and Blinn(Toon) for specular for all of the objects and character except the ground.

Toon + Phong

Vertex shader

calculates the general position, fragment position, fragment norm and uv coordinates given the inputs from uniform buffers in the cpp code. Then it gives the output to the fragment shader.

Fragment Shader

Used toon BRDF model for diffuse and Phong for specular for the ground.

Text

Vertex shader

calculates the general position, coordinates and color given the inputs from uniform buffers in the cpp code. Then it gives the output to the fragment shader.

Fragment Shader

Map the texture of fonts to each character including the input color which is an input for fragment shader

Overlay

Vertex shader

calculates the general position and uv coordinates given the inputs from uniform buffers in the cpp code. Then it gives the output to the fragment shader.

Fragment Shader

Just map the texture on the object regarding its uv coordinates

Uniform Blocks Data Structure

```
struct GlobalUniformBufferObject
{
	alignas(16) glm::vec3 DlightDir;
	alignas(16) glm::vec3 DlightColor;
	alignas(16) glm::vec3 AmbLightColor;
	alignas(16) glm::vec3 eyePos;
};

struct UniformBufferObject
{
	alignas(4) float visible;
	alignas(4) float amb;
	alignas(4) float gamma;
	alignas(16) glm::vec3 sColor;
	alignas(16) glm::mat4 mvpMat;
	alignas(16) glm::mat4 mMat;
	alignas(16) glm::mat4 nMat;
};

struct OverlayUniformBlock
{
	alignas(4) float visible;
};
```

Vertex Formats

```
struct VertexMesh
{
	glm::vec3 pos;
	glm::vec3 norm;
	glm::vec2 uv;
};

struct VertexOverlay
{
	glm::vec2 pos;
	glm::vec2 UV;
};
```

Descriptor Set Layouts

Variable	Binding	Type	Which shader
DSLGubo	0	UBO	ALL
DSLToon	0	UBO	ALL
	1	Texture	Fragment
DSLToonPhong	0	UBO	ALL
	1	Texture	Fragment
DSLOverlay	0	UBO	ALL
	1	Texture	Fragment

Vertex Descriptors

Variable	Format (C++)	Location	Type	Usage
VMesh	VertexMesh	0	vec3	POSITION
		1	vec3	NORMAL
		2	vec2	UV
VOverlay	VertexOverlay	0	vec2	OTHER
		1	Vec2	UV

Pipelines

Variable	Vertex Shader	Fragment Shader	Vertex format	Vertex descriptor	Set ID	Descriptor set Layout
PToon	ToonVert.spv	ToonFrag.spv	VertexMesh	VMesh	0	DSLGubo
					1	DSLToon
PToonPhong	ToonPhongVert.spv	ToonPhongFrag.spv	VertexMesh	VMesh	0	DSLGubo
					1	DSLToonPhong
POverlay	OverlayVert.spv	OverlayFrag.spv	VertexOverlay	VOverlay	0	DSLOverlay

Mesh Objects

Variable	Vertex Format	Vertex descriptor	Type	Model File
MCharacter	VertexMesh	VMesh	Manual	-
MGround	VertexMesh	VMesh	OBJ	ground.obj
MPlant	VertexMesh	VMesh	OBJ	plant.obj
MFlower	VertexMesh	VMesh	OBJ	flower.obj
MMountain	VertexMesh	VMesh	OBJ	mountain.obj
MSmallRock	VertexMesh	VMesh	OBJ	smallrock.obj
MStump	VertexMesh	VMesh	OBJ	stump.obj
MCloud	VertexMesh	VMesh	OBJ	cloud.obj
MRock	VertexMesh	VMesh	OBJ	rock2.obj

MTree	VertexMesh	VMesh	OBJ	tree.obj
MTree1	VertexMesh	VMesh	OBJ	tree1.obj
MTree2	VertexMesh	VMesh	OBJ	tree2.obj
MTree3	VertexMesh	VMesh	OBJ	tree3.obj
MTree4	VertexMesh	VMesh	OBJ	tree4.obj
MItem	VertexMesh	VMesh	MGCG	item.mgcg
MSpike	VertexMesh	VMesh	MGCG	spike.mgcg
MStartPanel	VertexOverlay	VOverlay	Manual	-
MEndPanel	VertexOverlay	VOverlay	Manual	-
MLosePanel	VertexOverlay	VOverlay	Manual	-
MInteraction Msg	VertexOverlay	VOverlay	Manual	-

Textures

Variable	File	Sampler
TCharacter	Character.jpg	-
TGround	Ground.png	-
TEnv	Texture_01.jpg	-
TEnv2	Texture_2.png	-
TItem	Textures_Dungeon.png	-
TStartPanel	Menu.jpg	-
TEndPanel	Ending.jpg	-
TLosePanel	LosePanel.png	-
TInteractionMsg	InteractMsg.png	-

Uniform Block Objects

Type	Variable
MeshUniformBlock	uboCharacter
MeshUniformBlock	uboGround
MeshUniformBlock	uboPlant
MeshUniformBlock	uboFlower
MeshUniformBlock	uboMountain
MeshUniformBlock	uboSmallRock
MeshUniformBlock	uboStump
MeshUniformBlock	uboCloud
MeshUniformBlock	uboRock
MeshUniformBlock	uboTree

Type	Variable
MeshUniformBlock	uboCharacter
MeshUniformBlock	uboGround
MeshUniformBlock	uboPlant
MeshUniformBlock	uboFlower
MeshUniformBlock	uboMountain
MeshUniformBlock	uboSmallRock
MeshUniformBlock	uboStump
MeshUniformBlock	uboCloud
MeshUniformBlock	uboRock
MeshUniformBlock	uboTree1
MeshUniformBlock	uboTree2
MeshUniformBlock	uboTree3
MeshUniformBlock	uboTree4
MeshUniformBlock	uboItem
MeshUniformBlock	uboSpike
OverlayUniformBlock	uboStartPanel
OverlayUniformBlock	uboEndPanel
OverlayUniformBlock	uboLosePanel
OverlayUniformBlock	uboInteractionMsg
GlobalUniformBlock	gubo

Descriptor Sets

Variable	Descriptor Set Layout	Binding	Type	C++ data structure	Variable with values	Texture
DSGubo	DSLGubo	0	UBO	GlobalUniformBlock	gubo	
DSCharacter	DSLToon	0	UBO	MeshUniformBlock	uboCharacter	
		1	Texture			TCharacter
DSGround	DSLToonPhong	0	UBO	MeshUniformBlock	uboGround	
		1	Texture			TGround
DSPlant	DSLToon	0	UBO	MeshUniformBlock	uboPlant	
		1	Texture			TEnv
DSFlower	DSLToon	0	UBO	MeshUniformBlock	uboFlower	
		1	Texture			TEnv

DSMountain	DSLToon	0	UBO	MeshUniformBlock	uboMountain	
		1	Texture			TEnv2
DSSmallRock	DSLToon	0	UBO	MeshUniformBlock	uboSmallRock	
		1	Texture			TEnv
DSStump	DSLToon	0	UBO	MeshUniformBlock	uboStump	
		1	Texture			TEnv
DSCloud	DSLToon	0	UBO	MeshUniformBlock	uboCloud	
		1	Texture			TEnv2
DSRock	DSLToon	0	UBO	MeshUniformBlock	uboRock	
		1	Texture			TEnv
DSTree	DSLToon	0	UBO	MeshUniformBlock	uboTree	
		1	Texture			TEnv2
DSTree1	DSLToon	0	UBO	MeshUniformBlock	uboTree1	
		1	Texture			TEnv
DSTree2	DSLToon	0	UBO	MeshUniformBlock	uboTree2	
		1	Texture			TEnv
DSTree3	DSLToon	0	UBO	MeshUniformBlock	uboTree3	
		1	Texture			TEnv
DSTree4	DSLToon	0	UBO	MeshUniformBlock	uboTree4	
		1	Texture			TEnv
DSItem	DSLToon	0	UBO	MeshUniformBlock	uboItem	
		1	Texture			TItem
DSSpike	DSLToon	0	UBO	MeshUniformBlock	uboSpike	
		1	Texture			TItem
DSStartPanel	DSLOverlay	0	UBO	OverlayUniformBlock	uboStartPanel	
		1	Texture			TStartPanel
DSEndPanel	DSLOverlay	0	UBO	OverlayUniformBlock	uboEndPanel	
		1	Texture			TEndPanel
DSLosePanel	DSLOverlay	0	UBO	OverlayUniformBlock	uboLosePanel	
		1	Texture			TLosePanel
DSInteraction Msg	DSLOverlay	0	UBO	OverlayUniformBlock	uboInteraction Msg	
		1	Texture			TInteraction Msg

Scene Objects

Name	Pipeline	Mesh	Descriptor Sets
Character	PToon	MCharacter	DSGubo
			DSCharacter
Ground	PToonPhong	MGround	DSGubo,
			DSGround
Plant	PToon	MPlant	DSGubo,
			DSPlant
Flower	PToon	MFlower	DSGubo
			DSFlower
Mountain	PToon	MMountain	DSGubo
			DSMountain
SmallRock	PToon	MSmallRock	DSGubo,
			DSSmallRock
Stump	PToon	MStump	DSGubo
			DSStump
Cloud	PToon	MCloud	DSGubo
			DSCloud
Rock	PToon	MRock	DSGubo
			DSRock
Tree	PToon	MTree	DSGubo
			DSTree
Tree1	PToon	MTree1	DSGubo
			DSTree1
Tree2	PToon	MTree2	DSGubo
			DSTree2
Tree3	PToon	MTree3	DSGubo
			DSTree3
Tree4	PToon	MTree4	DSGubo
			DSTree4
Item	PToon	MItem	DSGubo
			DSItem
Spike	PToon	MSpike	DSGubo
			DSSpike
Start Panel	POverlay	MStartPanel	DSStartPanel
End Panel	POverlay	MEndPanel	DSEndPanel
Lose Panel	POverlay	MLosePanel	DSLosePanel
Interaction Message	POverlay	MInteractionMsg	DSInteractionMsg