You need:

[] – 3D assets, in either OBJ or GLTF form

* SlotBody.obj – 1 instance - MBody
* City.obj – 1 instance - MCity
* Car.obj – 1 instance – Mcar
* Dessert\_pie.obj – 2 instances - Mpie
* Interior.obj – 1 instance - Mkitchen
* Character.obj – 1 instance – Mfarmer
* apple.obj – 3 instances - Mapple
* Dungeon.obj – 1 instance - Mdungeon
* courts.mgcg – 1 instance – Mcourt

[] – 3D assets dynamically generated in the code

* Intro Splash screen – quad – only normalized screen coordinates 2D – 1 instance - MKey
* Apple 1 splash retrieved – quad – only normalized screen coordinates 2D – 1 instance – Msplash
* Apple 2 splash retrieved - quad – only normalized screen coordinates 2D – 1 instance – Msplash2
* Apple 3 splash retrieved - quad – only normalized screen coordinates 2D – 1 instance – Msplash3
* Victory message screen - quad – only normalized screen coordinates 2D – 1 instance – Mwin
* Pie message screen - quad – only normalized screen coordinates 2D – 1 instance – Mwin2

[] – Textures associated with the models

* SlotBody.png – Tbody
* Layout\_1.png – TKey
* TexturesCity.png – TCity
* apple\_logo.jpg – TSplash
* TexturesInterior.png – Tkitchen
* Colors2.png – Tcar
* dessert\_pie.jpg – Tpie
* Colors2.png – Tfarmer
* TexturesDungeon.png – Tdungeon
* Wintokitchen.png – Twin
* Winzoom.png – Twin2

Then you decide:

[] – the illumination for the scene:

[] – which type of direct light? How many ?

Kitchen  
1) Direct light from the back

City  
1)Direct light from the back

Dungeon  
1) Point light in position (0,0,2)

Court  
1) Spot light

Pie Scene  
1) Three Spot lights with three different colours.

[] – Ambient light type?

Kitchen

* Constant Ambient

City

* Hemispheric Lighting

Dungeon

* Texture color multiplied by a coefficient

Court

* Texture color multiplied by a coefficient

Pie Scene

* Texture color multiplied by a coefficient

[] – Any object having emission?

No

* These terms might be enclosed in a scene-wide DataSetLayout
  + gubo DataSetLayout including:
    - Direct light color
    - Direct light position
    - Ambient light color
    - Viewer position
      * struct GlobalUniformBlock
  + DSLGubo1
    - 4 UNIFORM blocks including the data above
  + gubo DataSetLayout including:
    - 3 Direct light colors
    - 3 Direct light positions
    - Viewer position
      * struct PieUniformBlock
  + DSLGubo 5
    - 1 UNIFORM block including the data above
* For each asset

MBody, Mcity, Mpie, Mfarmer, Mkitchen, Mcar, Mapple, Mdungeon, Mcourt

* + [] – Define which vertex format it uses
    - Position
    - Normal vector
    - UV
      * Struct VertexMesh
  + [] – Select a BRDF approximation and shading technique, and depending on the scene illumination, define the corresponding Vertex / Fragment shader couple
    - Lambert + Blinn BRDF
  + [] – Decide which texture it requires
    - Color texture
  + [] – Decide which data sent from the CPP code the shaders need
    - Specular color
    - Specular power
    - Ambient sensitivity
    - World-view-projection matrix
    - World matrix
    - Normal transform matrix
      * struct MeshUniformBlock
    - The last two point determines the DataSetLayout for the shader couple
      * 1 UNIFORM block including the data above
      * 1 Texture with the corresponding color
        + DSLMesh1
* For each asset

MSplash, MSplash2, MSplash3, MKey, Mwin, Mwin2

* + [] – Define which vertex format it uses
    - Position (2D normalized screen coordinates)
    - UV
      * Struct VertexOverlay
  + [] – Select a BRDF approximation and shading technique, and depending on the scene illumination, define the corresponding Vertex / Fragment shader couple
    - No illumination, just pass the UV and return the pixel read from the texture
  + [] – Decide which texture it requires
    - Color Texture
  + [] – Decide which data sent from the CPP code the shaders need
    - Visibility
      * OverlayUniformBlock
    - The last two point determines the DataSetLayout for the shader couple
      * 1 Texture with the corresponding color
      * 1 UNIFORM block including the data above
        + DSLOverlay

You then:

[] – Examine how many different formats have been used by the assets

Two -> see above

* VMesh
* VOverlay

[] – How many different DataSetLayout are needed

* Three: DSLGubo1, DSLGubo5, DSLOverlay

[] – How many different vertex and fragment shaders are needed

* + This will also determine how many pipelines are needed

PMesh1

* + - Vertex Shader: MeshVert.spv
    - Fragment Shader: MeshFrag.spv
    - Based on VMesh1 and {DSLGubo1, DSLMesh1}

PMesh2

* + - Vertex Shader: MeshVert.spv
    - Fragment Shader: CityFrag.spv
    - Based on VMesh1 and {DSLGubo1, DSLMesh1}
  + PMesh3
    - Vertex Shader: MeshVert.spv
    - Fragment Shader: MeshFragCity2.spv
    - Based on VMesh1 and {DSLGubo1, DSLMesh1}
  + PMesh4
    - Vertex Shader: MeshVert.spv
    - Fragment Shader: BlinnFrag3.spv
    - Based on VMesh1 and {DSLGubo1, DSLMesh1}
  + PMesh5
    - Vertex Shader: MeshVert.spv
    - Fragment Shader: BlinnFrag5.spv
    - Based on VMesh1 and {DSLGubo1, DSLMesh1}
  + POverlay
    - Vertex Shader: OverlayVert.spv
    - Fragment Shader: OverlayFrag.spv
    - Based on VOverlay and {DSLOverlay}

You can then:

[] – Create the Vertex formats

[] – Define the models and load them

[] – Define the texture and load them

[] – Create a DataSetLayout for the scene-wide and pipeline specific uniform

[] – Create the pipelines needed

[] – For each scene-wide DataSetLayout, create the corresponding DataSet instance

* DSGubo1, DSGubo2, DSGubo3, DSGubo4 – instances DSLGubo1
  + struct GlobalUniformBlock
* DSGubo5 – instances DSLGubo5
  + struct PieUniformBlock
* DSBody, DScity, DSkitchen, DSapple1, DSapple2, DSapple3, DScar, DSpie, DSpie2, DSfarmer, DSdungeon, DScourt – instances of DSLMesh1
  + struct MeshUniformBlock
* DSSplash, DSSplash2, DSSplash3, DSKey, DSwin, DSwin2 – instance DSLOverlay
  + struct OverlayUniformBlcok

[] – Count the required number of:

* + DataSets: 22
  + DSGubo1, DSGubo2, DSGubo3, DSGubo4, DSGubo5, DSBody, DScity, DSapple1, DSapple2, DSapple3, DScar, DSpie, DSpie2, DSfarmer, DSdungeon, DScourt, DSsplash, DSsplash2, DSsplash3, DSkey, DSwin, DSwin2
* UniformBlocks elements of the DataSets: 22
  + All DS
* Texture elements of the DataSets: 17
  + All DS except DSGubo

[] – For each 3D asset, create its specific DataSet according to the corresponding DataSetLayout. Here is where you will define the size of the corresponding uniform, and assign the textures.

* Init the variables above

[] – In the procedure that populates the command buffer, enter the command to draw all the primitives:

[] – first bind the scene-wide DataSets

[] – for each different pipeline:

* + [] - Bind the pipeline
  + [] - For each object belonging to that pipeline:
    - [] – Bind the corresponding DataSet
    - [] – Bind the vertex and index buffers
    - [] – call the draw command for the corresponding mesh
* Remember: it is always easier to load all the 3D objects at the beginning, and then “hide” the ones you do not need by either giving them a zero scale, or by moving them far away from the far plane of the camera.

[] – initialize all the variables for the game logic

[] – in the procedure that handles the user interaction:

[] – Read the user input (from the keyboard, the mouse or the Joystick)

* + Orbiting camera model – left stick moves camera forward or up / down, right thumb moves the camera around the slot machine. Implented by storing the target position and the camera position and using a LookAt matrix
    - Four float variables needed: CamH, CamRadius, CamPitch, CamYaw
  + Implement the state machine of the game

[] – update the camera position and direction (if needed), and the corresponding view / projection matrix

* + Camera FoV = 90 deg, near plane = 0.1, far plane = 100

[] – update the variable with the position of the objects

* + Apple1Pos, Apple2Pos, Apple3Pos, cameraPos, Pie2Pos, Roll, Pitch, Yaw, CarPosition

[] – determine the new values of the uniform variable and map them

**1 - Vertex formats (C++)**

|  |  |
| --- | --- |
| **Name** | **Data structure** |
| VertexOverlay | struct VertexOverlay {  glm::vec2 pos;  glm::vec2 UV;  }; |
| VertexMesh | struct VertexMesh {  glm::vec3 pos;  glm::vec3 norm;  glm::vec2 UV;  }; |

**2 - Data structures for Uniform Block Objects (C++)**

|  |  |
| --- | --- |
| **Name** | **Data structure** |
| MeshUniformBlock | struct MeshUniformBlock {  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;  }; |
| OverlayUniformBlock | struct OverlayUniformBlock {  alignas(4) float visible;  }; |
| GlobalUniformBlock | struct GlobalUniformBlock {  alignas(16) glm::vec3 DlightDir;  alignas(16) glm::vec3 DlightColor;  alignas(16) glm::vec3 AmbLightColor;  alignas(16) glm::vec3 eyePos;  alignas(16) glm::vec3 lightPos;  alignas(16) glm::vec3 lightDir;  alignas(16) glm::vec4 lightColor;  }; |
| PieUniformBlock | struct PieUniformBlock {  alignas(16) glm::vec3 lightPos1;  alignas(16) glm::vec3 lightDir1;  alignas(16) glm::vec4 lightColor1;  alignas(16) glm::vec3 lightPos2;  alignas(16) glm::vec3 lightDir2;  alignas(16) glm::vec4 lightColor2;  alignas(16) glm::vec3 lightPos3;  alignas(16) glm::vec3 lightDir3;  alignas(16) glm::vec4 lightColor3;  alignas(16) glm::vec3 eyePos;  }; |

**3 - Data Set Layouts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Binding** | **Type** | **Which shader** |
| DSLMesh1 | 0 | ubo | All graphics |
| 1 | texture | Fragment |
|  |  |  |
| DSLOverlay | 0 | ubo | All graphics |
| 1 | texture | Fragment |
|  |  |  |
| DSLGubo1 | 0 | texture | Fragment |
|  |  |  |
|  |  |  |
| DSLGubo5 | 0 | texture | Fragment |
|  |  |  |
|  |  |  |

**4 - Vertex Descriptors**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Format (C++)** | **Location** | **Type** | **Usage** |
| VMesh1 | vertexMesh | 0 | Vec3 | position |
| 1 | Vec3 | norm |
| 2 | Vec2 | Texture coord |
| VOverlay | vertexOverlay | 0 | Vec3 | position |
| 1 | Vec2 | Texture coord |
|  |  |  |

**5 - Pipelines**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Variable** | **Vertex Shader** | **Fragment Shader** | **Vertex format (C++)** | **Vertex descriptor** | **Set ID** | **Data set Layout** |
| PMesh1 | MeshVert.spv | MeshFrag.spv | vertexMesh | VMesh1 | 0 | DSLGubo1 |
| 1 | DSLMesh1 |
|  |  |
| PMesh2 | MeshVert.spv | MeshFrag.spv | vertexMesh | VMesh1 | 0 | DSLGubo1 |
| 1 | DSLMesh1 |
|  |  |
| PMesh3 | MeshVert.spv | MeshFragCity2.spv | vertexMesh | VMesh1 | 0 | DSLGubo1 |
| 1 | DSLMesh1 |
|  |  |
| PMesh4 | MeshVert.spv | BlinnFrag3.spv | vertexMesh | VMesh1 | 0 | DSLGubo1 |
| 1 | DSLMesh1 |
|  |  |
| PMesh5 | MeshVert.spv | BlinnFrag5 | vertexMesh | VMesh1 | 0 | DSLGubo1 |
| 1 | DSLMesh1 |
|  |  |
| POverlay | OverlayVert.spv | OverlayFrag.spv | vertexOverlay | VOverlay | 0 | DSLOverlay |
|  |  |
|  |  |

**6 - Mesh objects**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Vertex Format (C++)** | **Vertex descriptor** | **Type** | **Model File** |
| MBody | VertexMesh | VMesh1 | obj | SlotBody.obj |
| Mcity | VertexMesh | VMesh1 | obj | City.obj |
| Mcar | VertexMesh | VMesh1 | obj | Car.obj |
| Mpie | VertexMesh | VMesh1 | obj | dessert\_pie.obj |
| Mkitchen | VertexMesh | VMesh1 | obj | Interior.obj |
| Mfarmer | VertexMesh | VMesh1 | obj | Character.obj |
| Mapple | VertexMesh | VMesh1 | obj | apple.obj |
| Mdungeon | VertexMesh | VMesh1 | obj | Dungeon.obj |
| Mcourt | VertexMesh | VMesh1 | MGCG | courts.mgcg |
| Mkey | VertexOverlay | VOverlay | manual |  |
| MSplash | VertexOverlay | VOverlay | manual |  |
| MSplash1 | VertexOverlay | VOverlay | manual |  |
| MSplash2 | VertexOverlay | VOverlay | manual |  |
| MSplash3 | VertexOverlay | VOverlay | manual |  |
| Mwin | VertexOverlay | VOverlay | manual |  |
| Mwin2 | VertexOverlay | VOverlay | manual |  |

**7 - Textures**

|  |  |  |
| --- | --- | --- |
| **Variable** | **File** | **Sampler** |
| TBody | SlotBody.png |  |
| Tkey | Layout\_1.png |  |
| TSplash | Apple\_logo.png |  |
| TCity | TextureCity.png |  |
| Tkitchen | TextureInterior.png |  |
| Tcar | Colors2.png |  |
| Tpie | Dessert\_pie.png |  |
| Tapple | Apple\_color.png |  |
| Tdungeon | TextureDungeon.png |  |
| Twin | Wintokitchen.png |  |
| Twin2 | Winzoom.png |  |

**8 - Uniform Blocks Objects, C++ sides**

|  |  |
| --- | --- |
| **Type** | **Variable** |
| MeshUniformBlock | uboBody |
| MeshUniformBlock | uboCity |
| MeshUniformBlock | ubocar |
| MeshUniformBlock | Uboapple1 |
| MeshUniformBlock | Uboapple2 |
| MeshUniformBlock | Uboapple3 |
| MeshUniformBlock | Ubokitchen |
| MeshUniformBlock | ubofarmer |
| MeshUniformBlock | ubopie |
| MeshUniformBlock | Ubopie2 |
| MeshUniformBlock | Ubodungeon |
| MeshUniformBlock | Ubocourt |
| GlobalUniformBlock | Gubo1 |
| GlobalUniformBlock | Gubo2 |
| GlobalUniformBlock | Gubo3 |
| GlobalUniformBlock | Gubo4 |
| OverlayUniformBlock | uboKey |
| OverlayUniformBlock | uboSplash |
| OverlayUniformBlock | uboSplash2 |
| OverlayUniformBlock | uboSplash3 |
| OverlayUniformBlock | uboWin |
| OverlayUniformBlock | uboWin2 |
| PieUniformBlock | Gubo5 |

**9 - Data Sets**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Variable** | **Data Set Layout** | **Binding** | **Type** | **C++ data structure** | **Variable with values** | **Texture** |
| DSBody | DSLMesh1 | 0 | UBO | MeshuniformBlock | uboBody |  |
| 1 | tex |  |  | TBody |
|  |  |  |  |  |
| DSKey | DSLOverlay | 0 | UBO | OverlayuniformBlock | ubokey |  |
| 1 | tex |  |  | TKey |
|  |  |  |  |  |
| DSSplash | DSLOverlay | 0 | UBO | OverlayuniformBlock | uboSplash |  |
| 1 | tex |  |  | TSplash |
|  |  |  |  |  |
| DSSplash2 | DSLOverlay | 0 | UBO | OverlayuniformBlock | uboSplash2 |  |
| 1 | tex |  |  | TSplash |
|  |  |  |  |  |
| DSSplash3 | DSLOverlay | 0 | UBO | OverlayuniformBlock | uboSplash3 |  |
| 1 | tex |  |  | TSplash |
|  |  |  |  |  |
| DSwin | DSLOverlay | 0 | UBO | OverlayuniformBlock | ubowin |  |
| 1 | tex |  |  | Twin |
|  |  |  |  |  |
| DSwin2 | DSLOverlay | 0 | UBO | OverlayuniformBlock | Ubowin2 |  |
| 1 | tex |  |  | Twin2 |
|  |  |  |  |  |
| DSGubo1 | DSLGubo1 | 0 | UBO | GlobaluniformBlock | Gubo1 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| DSGubo2 | DSLGubo1 | 0 | UBO | GlobaluniformBlock | Gubo2 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| DSGubo3 | DSLGubo1 | 0 | UBO | GlobaluniformBlock | Gubo3 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| DSGubo4 | DSLGubo1 | 0 | UBO | GlobaluniformBlock | Gubo4 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| DSGubo5 | DSLGubo5 | 0 | UBO | PieluniformBlock | Gubo5 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| DSCity | DSLMesh1 | 0 | UBO | MeshuniformBlock | ubocity |  |
| 1 | tex |  |  | TCity |
|  |  |  |  |  |
| DSkitchen | DSLMesh1 | 0 | UBO | MeshuniformBlock | ubokitchen |  |
| 1 | tex |  |  | Tkitchen |
|  |  |  |  |  |
| DSapple1 | DSLMesh1 | 0 | UBO | MeshuniformBlock | Uboapple1 |  |
| 1 | tex |  |  | Tapple |
|  |  |  |  |  |
| DSapple2 | DSLMesh1 | 0 | UBO | MeshuniformBlock | Uboapple2 |  |
| 1 | tex |  |  | Tapple |
|  |  |  |  |  |
| DSapple3 | DSLMesh1 | 0 | UBO | MeshuniformBlock | Uboapple3 |  |
| 1 | tex |  |  | Tapple |
|  |  |  |  |  |
| DScar | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubocar |  |
| 1 | tex |  |  | Tcar |
|  |  |  |  |  |
| DSpie | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubopie |  |
| 1 | tex |  |  | Tpie |
|  |  |  |  |  |
| DSpie2 | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubopie2 |  |
| 1 | tex |  |  | Tpie |
|  |  |  |  |  |
| DSfarmer | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubopie |  |
| 1 | tex |  |  | Tpie |
|  |  |  |  |  |
| DSdungeon | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubodungeon |  |
| 1 | tex |  |  | Tdungeon |
|  |  |  |  |  |
| DScourt | DSLMesh1 | 0 | UBO | MeshuniformBlock | Ubocourt |  |
| 1 | tex |  |  | Tcourt |
|  |  |  |  |  |

**10 - Scene Objects**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Pipeline** | **Mesh** | **Data Set** |
| The body of the machine | PMesh2 | MBody | DSbody |
| The city | PMesh2 | MCity | DSCity |
| The kitchen | PMesh1 | Mkitchen | DSKitchen |
| The apple1 | PMesh2 | Mapple | DSapple1 |
| The apple2 | PMesh3 | Mapple | DSapple2 |
| The apple3 | PMesh4 | Mapple | DSapple3 |
| The car | PMesh2 | Mcar | DScar |
| The pie1 | PMesh5 | Mpie | DSpie |
| The pie2 | PMesh1 | Mpie | DSpie2 |
| The farmer | PMesh1 | Mfarmer | DSfarmer |
| The dungeon | PMesh3 | Mdungeon | DSdungeon |
| The court | PMesh4 | Mcourt | DScourt |