**Creating Virtual Worlds on the GPU**

**CIS6015**

**Ben Thorpe st20119834**

**Question 1:**

Graphics Pipeline

Graphical Processing Units (GPUs) are based around a pipeline structure. Describe each stage of the pipeline and how it processes data from geometry at the front-end to pixels at the backend. As part of your answer, identity which stages are programmable and which are not programmable and explain why. **(20 Marks)**

The graphical pipeline is a fixed linear process requiring data to be pass through from each stage. The pipeline doesn’t allow for data to be inputted mid stage. This means that all the required data must be inputted at the start and pass through. This results in a pass-through shader program that simple passes data through the stage while not altering the data in anyway. The stages start with the vertex data, to the vertex shader and geometry shader, perspective transformation, rasterization, fragment shader and then to the framebuffer to be rendered.

The process starts with the object data that is passed to the vertex stage. The vertex shader creates the vertexes with coordinates. The vertex shader allows alterations using programs that changes the vertex colour and texture position. Using a vertex shader allows for changing texture mapping coordinates, moving the vertex.

The next stage is the geometry shader which uses the vertexes to create polygonal objects. This is an optimal stage that when skipped the vertexes are simply passed to the next stage. The geometry stage allows for programmers to alter the shape and geometry for the object. This includes creating new geometry for objects. Commonly grass and foliage are created using geometry shaders, as this saves storage the data on the CPU. The main reason for the optimal natures is the negative impact on the speed, massive slowing the rendering process.

The next stage is perspective transformation and clipping, at this point the object is imported into the scene with respect to the camera matrix. The object size and rotation are factored creating a relative scene model. In addition, overlapping models, creates clipping effects. This stage is not programmable and is handled by the rendering system. The main reason for the lack of programmable is that the desired functions can be performed using vertex and geometry shaders. Objects can be moved, altered using the shaders previous. Furthermore, the code is much more complex with advance 3-dimensional Maths, and it is simple easier for the data to be manipulated before using both the vertex shader and geometry shader.

The next shader is rasterization and colorization. This stage converts the model data to pixels and assign colour to the object. The process takes the vertex points and maps them to the screen as pixels. Then the area in between each point is calculated and colours. This stage again contains complex code and is not open to programming, due to the complex nature and manipulation of colour and texture can be done using the fragment shader.

The fragment shaders allow for programmers to manipulate colour and textures attached to objects. The current colour and texture coordinates are calculated be the rasterization and given to the fragment shader to be manipulated. Lastly the data is passed to the frame buffer where the rendering system will render the data that is there.

Meta-balls

1. Describe in detail the meta-balls algorithm and how it may be used to simulate the merging of water droplets. As part of your answer, discuss the strengths and weakness of using meta-balls compared to using standard polygonal structures to animate this effect. **(10 Marks)**

Meta-balls or blobby surface is an alternative method of storing model data that uses functions to describe shapes rather than vertex data. An object has a field in which it controls and effects, this is created by the field function. Instead of fixed points create creates polygons the function is calculated at runtime allowing for fluid animation and alteration. To simulate water droplets, each single droplet would be containing a single positional center and the object would be created using a field function. The function was determining the shape of the droplet. The field function calculates using control points to determine the field strength.

The first noticeable strength of meta balls is the fluid animation and modelling. Since meta-balls calculate the model structure during run-time this allows for constant alterations. Furthermore, since the model is defined as a function, manipulating the feature will alter the shape, where using polygons would require changing the vertex data constantly. This results in easier and simple alterations using meta-balls as only a function is needed where entire massive lists of vertex data would have to be altered. This makes meta-balls more suitable to produce fluid models.

The meta-ball functions calculate fields strength that can be altered by external factors. Other rain droplets will alter the models of other droplets by simple existing. The function to calculated field strengths accords for other objects and alters accordingly. This allows for addition of external factors such as wind, collisions, merging of water droplets.

The biggest flaw of meta-balls is the required computational power that is needed. As each object is calculated at run-time, the CPU must perform countless algorithms to calculate the model data. These algorithms must be performed for each meta-ball object thus creating a problem with scalability as more models requires more calculations.

1. Describe the process required to calculate the unit normal vector on the surface of a blobby object. **(4 Marks)**

Since the models are calculated at run time, normal vectors must also be calculated in the same manner. Once the model has been created, to calculate the unit normal, first the unit normal will only applies to an individual surface point and an object will have multiple Normals at different points. To function, simply involves creating a vector between the selected surface point, towards the center of the meta ball. That vector is then normalized creating a unit vector.

1. Explain why ray tracing meta-balls is computationally very expensive, as screen resolutions increase. Include within your answer suitable approaches that may be taken to minimise the impact of ray-tracing meta-balls. **(6 Marks)**

Ray tracing involves incrementally stepping along a path within the game world and at each interval a check must be made. This check determines if a 3D point is part of an object, this requires checking the distance between the current point and all control point in the scene. Check all control points is the biggest calculate and as the bigger screen, meaning that the ray is longer, and more points are therefore checked against each control points. To reduce the computational cost, lowering the number of control points that are being check would help reduce the issue. By creating a surface area of the shape, an iso-surface, using this surface, once a ray is inside an iso-surface, there is no need to perform the ray check until the ray leaves the surface, this reduces the amount of ray checks needed especially for larger meta-balls as more area are being ignored, inversely smaller meta-balls would not benefit much from this approach. Furthermore, calculating the iso-surface requires computational resources.

**Question 5:**

Marching Cubes Algorithm

Explain, with aid of a suitable example, how the Marching Cubes algorithm can be used to render a 3D implicit surface. As part of your answer, critically evaluate the advantages and disadvantages of using Marching Cubes in comparison to ray-casting and justify why you might choose this approach in the context of real-time graphics rendering.

**(20 Marks)**

The marching cube algorithm is used to approximate the surface of an implicit object. The first step is to divide the screen into segments, using small cubes that fills the entire scene, the smaller the cubes the higher resolution at the cost of high computational cost. Then for each cube perform a check that determines the surface. The check itself involves analyses the corners of the cube and where or not there are in the object. Comparison the combinate of the corners, using a look up table as refence a line part is selected. For example, if the only the bottom corner is within the object, then the program selected a small bottom right corner line. For a 2D shape there are 16 possible combinations each with a unique line which are inserted. the midpoint of the cube. The next step is connecting all the lines to create a shape that describes the surface. The same approach is done within 3D just instead of 4 corners it uses all 8 corners of the cube, and the replacement line is a more complex piece of geometry that when combines creates an iso-surface around the 3D object.

The first strength of marching cube is the relatively low computational cost. Since the only calculates involves looking at corners and comparing that with a look-up table. This algorithm is not expensive as there is only 15 combinations to look at, eliminating rotations, this is a small list to look through and compare. This is less expensive of an algorithm then ray casting and performing distance check using control points. At being said larger objects will result in larger volumes being ignored. As the volume is within the iso-surface therefore ray casts within the object are redundant, as the system already knows the ray is within the surface. This reduces the amounts of checks that are performed. When rendering the system will use the iso-surface and assume that every point within the iso-surface is a part of the is-surface therefore the system doesn’t need to check each point within the iso-surface.

Inversely using marching cubes for smaller objects is less effective as the smaller iso-surface results in less volume being ignored. While less points are being ignored, it still reduces computational costs as control points are being ignored and there are less ray checks points performed. As the objects get smaller cubes are required, this creates a higher resolution, but requires more resources to calculate the additional cubes. Furthermore, according to the look up table an object that is smaller than the cubes are ignored therefore creating a size limit of objects. However, ray casting doesn’t help either with smaller objects or reducing the size of the cubes will still be more efficient than ray casting.