# Tessellation & Interpolation

## EDAF80: Computer Graphics

Rikard Olajos

# AGENDA

1. C++ arrays
2. Tessellation
3. Interpolation
4. Assignment 2

# RAW ARRAYS: STACK & HEAP ALLOCATION

## Stack

```
float numbers[3];
numbers[0] = 1.0f;
...
```

## Stack: direct initialization

```
float numbers[3] = { 1.0f, 2.0f, 3.0f };
```

## Heap

```
float* numbers = new float[3];
number[0] = 1.0f;
...
delete[] numbers;
```

# STL ARRAYS: VECTOR & ARRAY

## Includes

```
#include <array>
#include <vector>
```

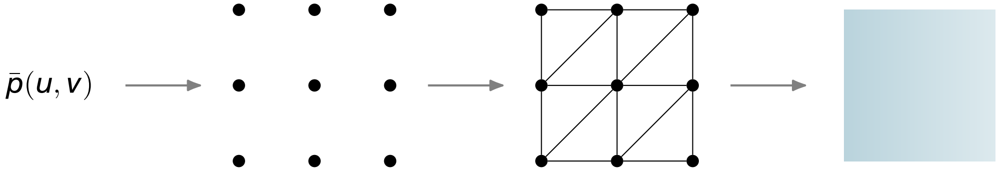
## Initialization

```
std::array<int, 3> arr;      // Static array with 3 integers
std::vector<float> vec;      // Dynamic array with floats (on the heap)
```

## Element access & size

```
arr[0] = 1;                  // Set first element to 1
vec.push_back(1.0f);         // Add 1.0f to end of vector
std::cout << vec[0];         // Print first element of 'vec'
std::cout << vec.size();     // Print number of elements in 'vec'
```

# TESSELLATION




1. Setup ***vertex array***
2. Setup ***index array*** (triangulate)

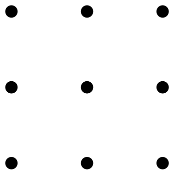
# CREATE VERTEX ARRAY

- Create vertex array (e.g. $3 \times 3$ vertices)

```
auto vertices = std::vector<glm::vec3>(9);
```

- Assign vertex

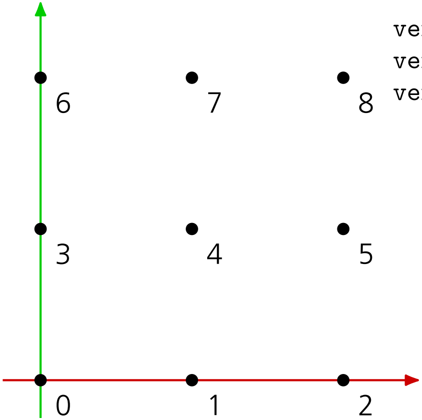
```
vertices[index] = glm::vec3(x, y, z);
```



# VERTEX ARRAY LAYOUT

```
vertices[8].x = 1.0;
vertices[8].y = 1.0;
vertices[8].z = 0.0;
```

```
vertices[3].x = 0.0;
vertices[3].y = 0.5;
vertices[3].z = 0.0;
```



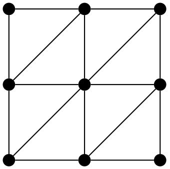
# TRIANGULATION

- Create index array ($2 \times (3-1)(3-1)$ triangles)

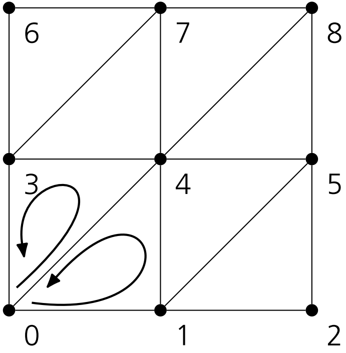
```
auto indices = std::vector<glm::uvec3>(8);
```

- Define triangle (indices for the three vertices)

```
indices[index] = glm::uvec3(v0, v1, v2);
```



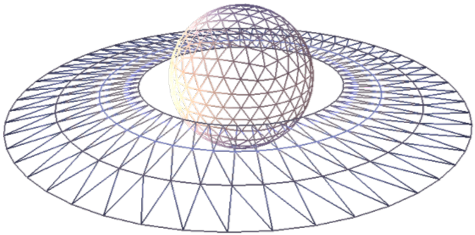
# INDEX ARRAY LAYOUT




- Indices in counter-clockwise order (`CCW`)
- Backface culling is off by default, turn it on to improve performance!

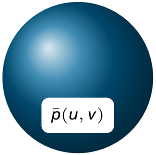
```
indices[0].x = 0;
indices[0].y = 1;
indices[0].z = 4;

indices[1].x = 0;
indices[1].y = 4;
indices[1].z = 3;
```

# PARAMETRIC SURFACES



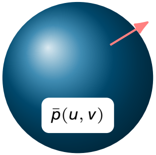
# PARAMETRIC SURFACE & TANGENT SPACE

- Map surface from 2D: $\bar{p}(x, y, z) = \bar{p}(u, v)$
- $\mathbb{R}^2 \mapsto \mathbb{R}^3$



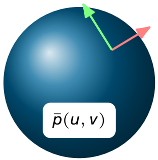
# PARAMETRIC SURFACE & TANGENT SPACE

- Map surface from 2D: $\bar{p}(x, y, z) = \bar{p}(u, v)$
- $\mathbb{R}^2 \mapsto \mathbb{R}^3$
  - Tangent $t = \frac{\partial \bar{p}}{\partial u}$



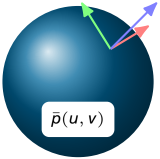
# PARAMETRIC SURFACE & TANGENT SPACE

- Map surface from 2D: $\bar{p}(x, y, z) = \bar{p}(u, v)$
- $\mathbb{R}^2 \mapsto \mathbb{R}^3$
  - Tangent $t = \frac{\partial \bar{p}}{\partial u}$
  - Binormal $b = \frac{\partial \bar{p}}{\partial v}$



# PARAMETRIC SURFACE & TANGENT SPACE

- Map surface from 2D: $\bar{p}(x, y, z) = \bar{p}(u, v)$
- $\mathbb{R}^2 \mapsto \mathbb{R}^3$
  - Tangent $t = \frac{\partial \bar{p}}{\partial u}$
  - Binormal $b = \frac{\partial \bar{p}}{\partial v}$
  - Normal $n = \frac{\partial \bar{p}}{\partial u} \times \frac{\partial \bar{p}}{\partial v}$

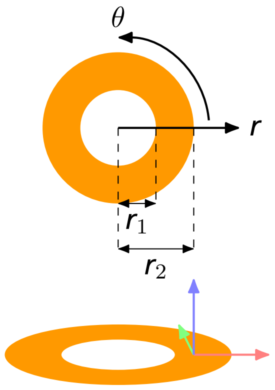


# CIRCLE RING

$$\bar{p}(r,\theta) = \begin{Bmatrix} r\cos(\theta) \\ r\sin(\theta) \\ 0 \end{Bmatrix} \text{ for } \begin{matrix} r_1 \leq r \leq r_2 \\ 0 \leq \theta < 2\pi \end{matrix}$$

$$t = \frac{\partial \bar{p}}{\partial r} = \begin{Bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{Bmatrix}$$

$$b = \frac{\partial \bar{p}}{\partial \theta} = \begin{Bmatrix} -r\sin(\theta) \\ r\cos(\theta) \\ 0 \end{Bmatrix}$$

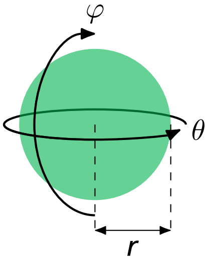


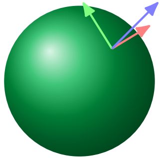
# SPHERE

$$\bar{p}(\theta,\varphi) = \begin{Bmatrix} r\sin(\theta)\sin(\varphi) \\ -r\cos(\varphi) \\ r\cos(\theta)\sin(\varphi) \end{Bmatrix} \text{ for } \begin{matrix} 0 \leq \theta \leq 2\pi \\ 0 \leq \varphi \leq \pi \end{matrix}$$

$$t = \frac{\partial\bar{p}}{\partial\theta} = \begin{Bmatrix} r\cos(\theta)\sin(\varphi) \\ 0 \\ -r\sin(\theta)\sin(\varphi) \end{Bmatrix}$$

$$b = \frac{\partial\bar{p}}{\partial\varphi} = \begin{Bmatrix} r\sin(\theta)\cos(\varphi) \\ r\sin(\varphi) \\ r\cos(\theta)\cos(\varphi) \end{Bmatrix}$$



## SPHERE

$$\bar{p}(\theta, \varphi) = \begin{Bmatrix} r\sin(\theta)\sin(\varphi) \\ -r\cos(\varphi) \\ r\cos(\theta)\sin(\varphi) \end{Bmatrix} \text{ for } \begin{matrix} 0 \leq \theta \leq 2\pi \\ 0 \leq \varphi \leq \pi \end{matrix}$$

$$t = \frac{\partial \bar{p}}{\partial \theta} = \begin{Bmatrix} r\cos(\theta)\sin(\varphi) \\ 0 \\ -r\sin(\theta)\sin(\varphi) \end{Bmatrix}$$

$$b = \frac{\partial \bar{p}}{\partial \varphi} = \begin{Bmatrix} r\sin(\theta)\cos(\varphi) \\ r\sin(\varphi) \\ r\cos(\theta)\cos(\varphi) \end{Bmatrix}$$

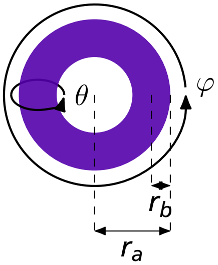
- $t$ and $b$ can be simplified since only direction is important
- $t$ ***needs*** to be simplified, or it will be undefined for $\varphi = 0$

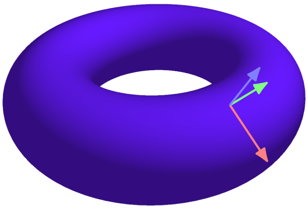
# TORUS

$$\bar{p}(\theta, \varphi) = \begin{Bmatrix} (r_a + r_b \cos(\theta)) \cos(\varphi) \\ (r_a + r_b \cos(\theta)) \sin(\varphi) \\ -r_b \sin(\theta) \end{Bmatrix} \text{ for } \begin{matrix} 0 \leq \theta \leq 2\pi \\ 0 \leq \varphi \leq 2\pi \end{matrix}$$

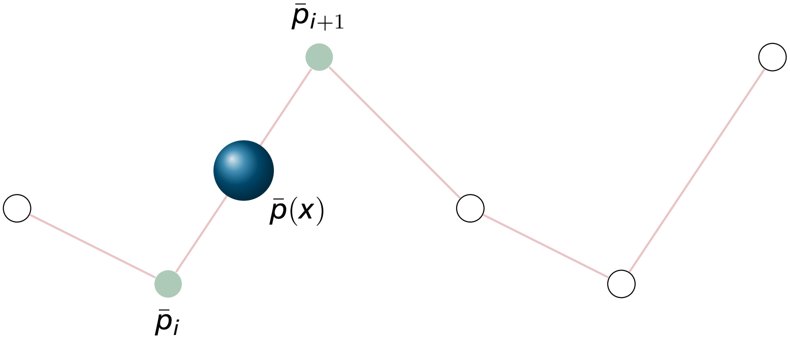
$$t = \frac{\partial \bar{p}}{\partial \theta} = \begin{Bmatrix} -r_b \sin(\theta) \cos(\varphi) \\ -r_b \sin(\theta) \sin(\varphi) \\ -r_b \cos(\theta) \end{Bmatrix}$$

$$b = \frac{\partial \bar{p}}{\partial \varphi} = \begin{Bmatrix} -(r_a + r_b \cos(\theta)) \sin(\varphi) \\ (r_a + r_b \cos(\theta)) \cos(\varphi) \\ 0 \end{Bmatrix}$$



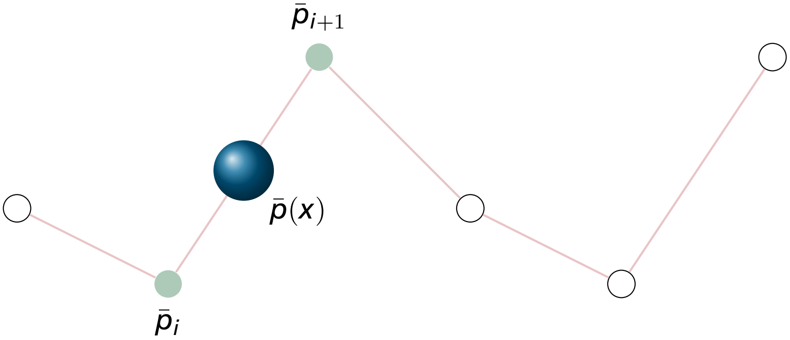



# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad \mathit{for} \quad x \in [0, 1]$$

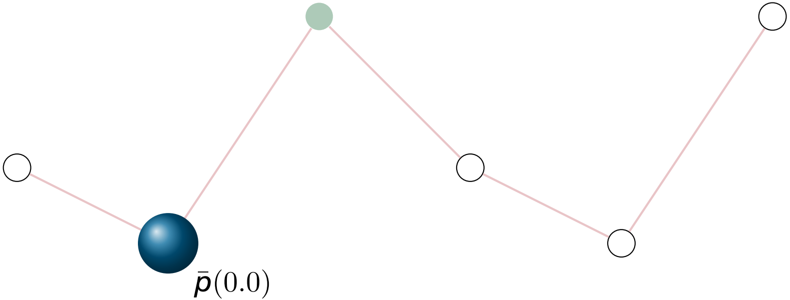
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad for \;\; x \in [0, 1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

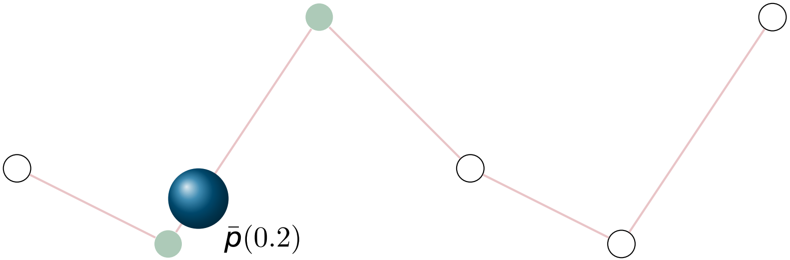
# LINEAR INTERPOLATION (LERP)



$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad for \quad x \in [0,1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

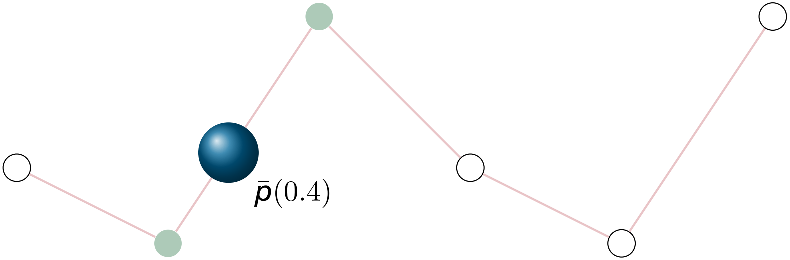
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad \textit{for} \quad x \in [0, 1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

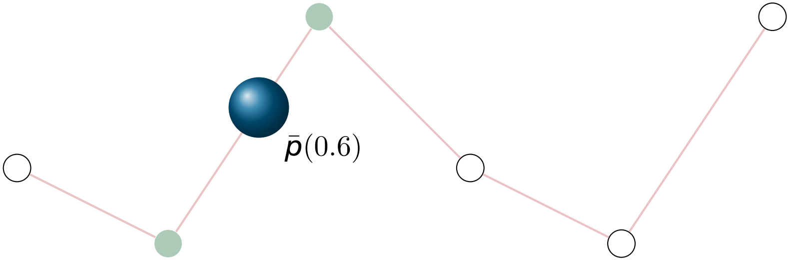
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad for \;\; x \in [0, 1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

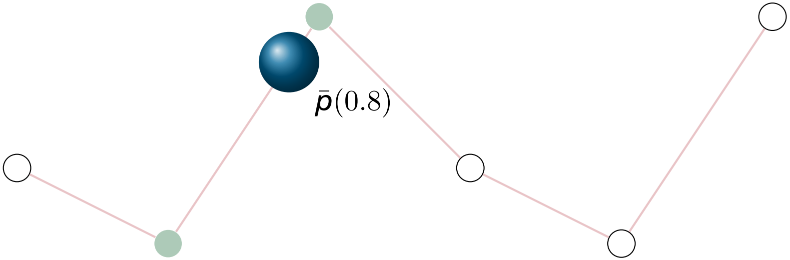
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad \mathit{for} \quad x \in [0, 1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

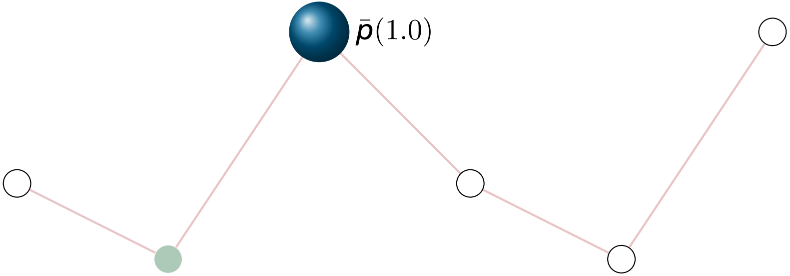
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad \mathit{for} \quad x \in [0, 1]$$

$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

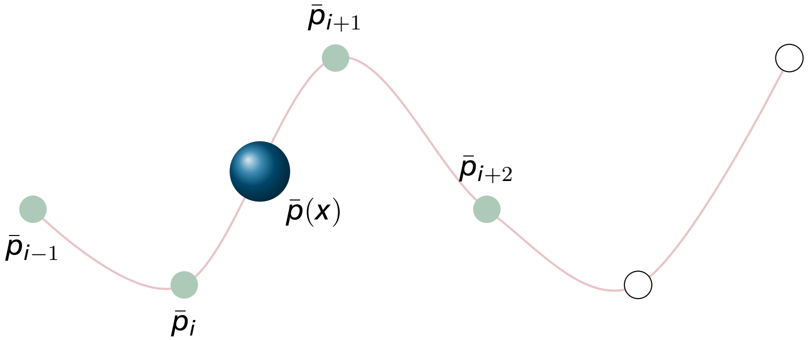
# LINEAR INTERPOLATION (LERP)




$$\bar{p}(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{p}_i \\ \bar{p}_{i+1} \end{bmatrix} \quad for \;\; x \in [0, 1]$$

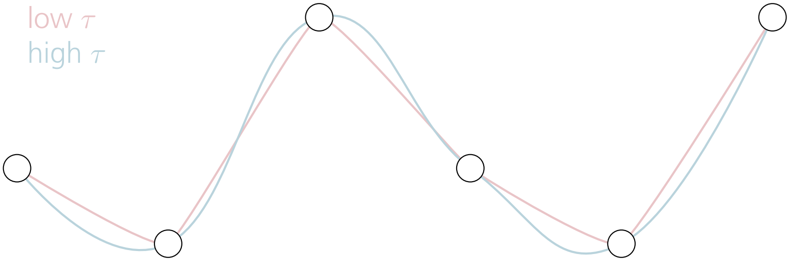
$$\bar{p}(x) = (1 - x)\bar{p}_i + x\bar{p}_{i+1}$$

# CUBIC INTERPOLATION (CATMULL-ROM)



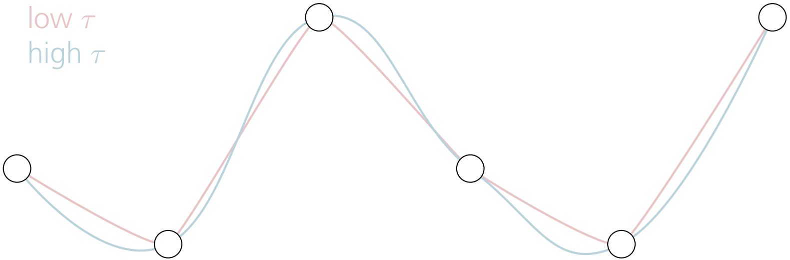

$$\bar{q}(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \bar{p}_{i-1} \\ \bar{p}_i \\ \bar{p}_{i+1} \\ \bar{p}_{i+2} \end{bmatrix} \quad \mathit{for} \;\; x \in [0, 1]$$

## TENSION




$$\bar{q}(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \bar{p}_{i-1} \\ \bar{p}_i \\ \bar{p}_{i+1} \\ \bar{p}_{i+2} \end{bmatrix} \quad \textit{for} \quad x \in [0, 1]$$

# TENSION




$$\bar{q}(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \bar{p}_{i-1} \\ \bar{p}_i \\ \bar{p}_{i+1} \\ \bar{p}_{i+2} \end{bmatrix} \quad \textit{for} \quad x \in [0, 1]$$

- $\tau =$ tension, how “stiff” the curve is at the control points
- Keep within $[0, 1]$
- Good initial value: $0.5$

# ASSIGNMENT 2

- Tessellate objects from parametric equations
- Linear and cubic interpolation
- Files you have to modify
  - `src/EDAF80/assignment2.cpp`
  - `src/EDAF80/parametric_shapes.cpp`
  - `src/EDAF80/interpolation.cpp`

# TESSELLATION

- Implement function bodies in `src/EDAF80/parametric_shapes.cpp`

```
bonobo::mesh_data parametric_shapes::createQuad(...);
bonobo::mesh_data parametric_shapes::createSphere(...);
bonobo::mesh_data parametric_shapes::createTorus(...); // Optional
```

# TESSELLATION

- Implement function bodies in `src/EDAF80/parametric_shapes.cpp`

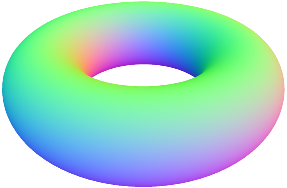
```
bonobo::mesh_data parametric_shapes::createQuad(...);
bonobo::mesh_data parametric_shapes::createSphere(...);
bonobo::mesh_data parametric_shapes::createTorus(...); // Optional
```

- Look at `createCircleRing(...)` in the same file for guidance
- Make sure parameter definitions and ranges are correct
  - Circle ring: $0 \leq \theta < 2\pi, \boldsymbol{r}_1 \leq \boldsymbol{r} \leq \boldsymbol{r}_2$
  - Sphere: $0 \leq \theta \leq 2\pi, 0 \leq \varphi \leq \pi$

# DEBUGGING NORMALS

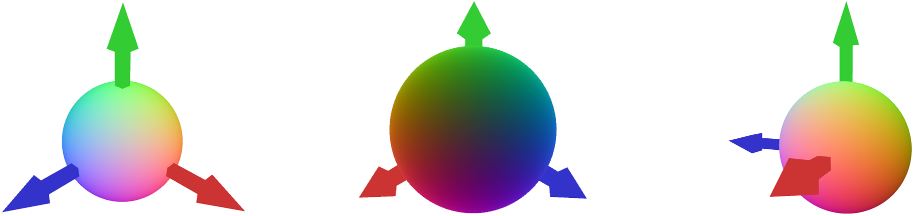
- Colourize, use the "Normals" shader to represent normals as RGB values

- Inspect illumination, is illumination consistent with the location of the light source?



# COLOURIZING NORMALS

- Map from $[-1,1]$ to $[0,1]$
  - $(\boldsymbol{N} \cdot 0.5) + 0.5$
- Example: Z axis $(0,0,1)$ becomes $(0.5,0.5,1)$
- Values are normalized: $(1,1,1) \mapsto (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$



$(1,1,1) \mapsto (1,1,1)$ $\quad$ $(-1,-1,-1) \mapsto (0,0,0)$ $\quad$ $(1,0,0) \mapsto (1,0.5,0.5)$

# TOOLS IN GUI

- Change cull mode: Disabled, Back faces, Front faces
- Change polygon mode: Fill, Line, Point
- Change shaders: Fallback, Diffuse, Normal, Tangent, Bitangent, Texture coords

# INTERPOLATION

- Implement linear and cubic interpolation
- Implement function bodies in `src/EDAF80/interpolation.cpp`

```
glm::vec3 interpolation::evalLERP(...);
glm::vec3 interpolation::evalCatmullRom(...);
```

- Test with just 2 (LERP) or 4 (cubic) points first
- Animate an object along the path using both function and the predefined control points
- `use_linear` and `catmull_rom_tension` variables are bound to the GUI and should be used

# IMPLEMENTATION SKETCH

```
// Init:
std::array<glm::vec3, N> control_points = { ... };
float path_pos = 0.0f;
float velocity = ...

// Main loop:
    int i = floor(path_pos);

    // Pick indices for interpolation: i-1, i, i+1, i+2

    // Make sure indices wrap: 0, 1, ..., N-1, 0, 1, ...

    // Call interpolation function with points from control_points

    path_pos += velocity;
```

# DEMO