Introduction

Edwin Chan

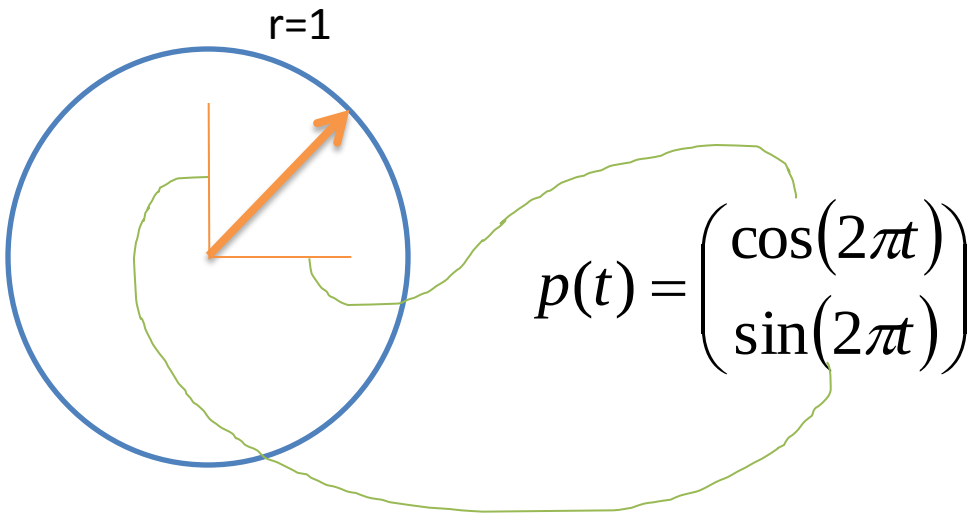
- Office: Room 571 “3D Vision Lab”
- Office Hours:
  - Mon 1pm – 3pm
  - Thu 12pm – 2 pm
  - Open Doors policy
- E-mail: ycha171@aucklanduni.ac.nz

Parametric Curves and Surfaces

- Describe the x, y, z coordinates by other parameters
- 2D curve: $$p(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$
  - where x(t) and y(t) are simple functions
  - t goes from 0 to 1

Parametric Curves and Surfaces

- e.g. Circle



- We can compute the tangent and normal at any point
  - Tangent:

$$p'(t) = \begin{pmatrix} \left(\frac{\partial x}{\partial t}\right) \\ \left(\frac{\partial y}{\partial t}\right) \end{pmatrix} = \begin{pmatrix} -2\pi \sin(2\pi t) \\ 2\pi \cos(2\pi t) \end{pmatrix}$$

  - Normal is perpendicular to tangent

Parametric Curves and Surfaces

- 3D surfaces:
  - One more parameter
  - Normal: Same idea, but the normal is perpendicular to both tangents
    - Find using the cross product

$$n(s,t) = \frac{\partial p}{\partial t} \times \frac{\partial p}{\partial s}$$

Surfaces of Revolution

- Form a 3D surface by rotating a 2D parametric curve

- e.g. Forming a disc by rotating a line
- e.g. Forming a sphere by rotating a semi-circle on x-z plane (written)

Surfaces of Revolution

- Preparing it:

```
for (int iT = 0; iT < MAX_T; iT++)
{
        float fT = (float) iT / (MAX_T – 1); // fT=[0,1]
        float fX = /* some x(t) */
        float fZ = /* some z(t) */

        for (int iS = 0; iS < MAX_S; iS++)
        {
                // fS=[0,2PI]
                float fS = 2*Pi * (float) iS / (MAX_S – 1);
                vertex[iT][iS][0] = fX
                vertex[iT][iS][1] = fZ * cos(fS);
                vertex[iT][iS][2] = fZ * sin(fs);
        }
}
```

Surfaces of Revolution

- Drawing it using GL_QUAD_STRIP:

```
for (int iT = 0; iT < (MAX_T-1); iT++)
{
        glBegin(GL_QUAD_STRIP);

        for (int iS = 0; iS <= MAX_S; iS++)
        {
                glVertex3fv(vertex[iT+1][iS%MAX_S]);
                glVertex3fv(vertex[iT][iS%MAX_S]);
        }

        glEnd();
}
```

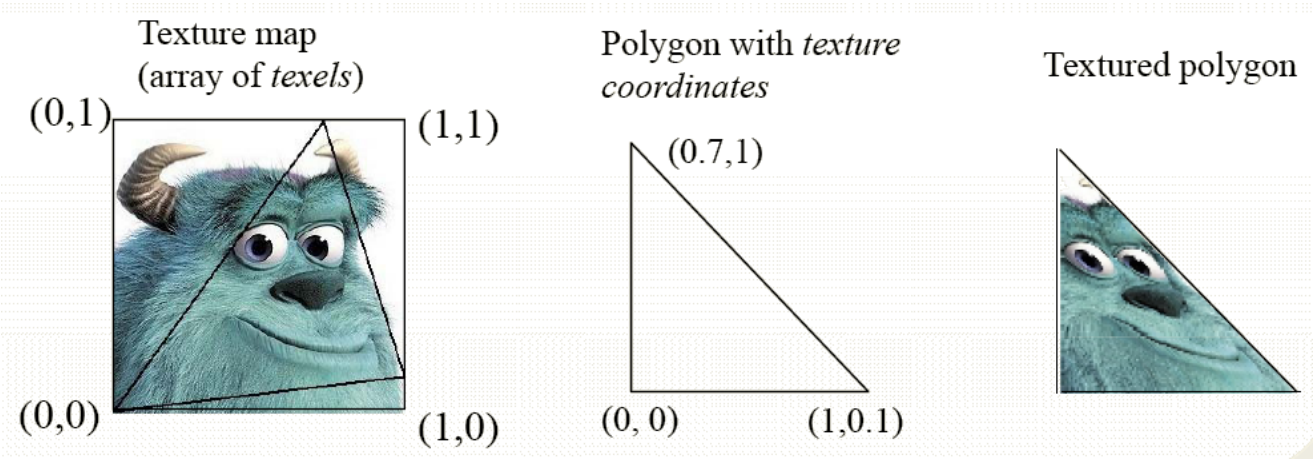
Texture Mapping in OpenGL

- Setup texture using glGenTextures(), glBindTexture(), glTexParameteri() and glTexImage2D(). (see lec notes)
- On drawing, simply add glTexCoord2f() before every glVertexf() to specify the texture coordinate of that vertex.

```
void display()
{
     ... // cut
     glBegin( some_mode );
     for (...) // loop through your object's vertices depending on mode
     {
          glTexCoord2f( some_texture_coord );
          glVertex3fv( vertex );
     }
     glEnd();
     ... /cut
}
```

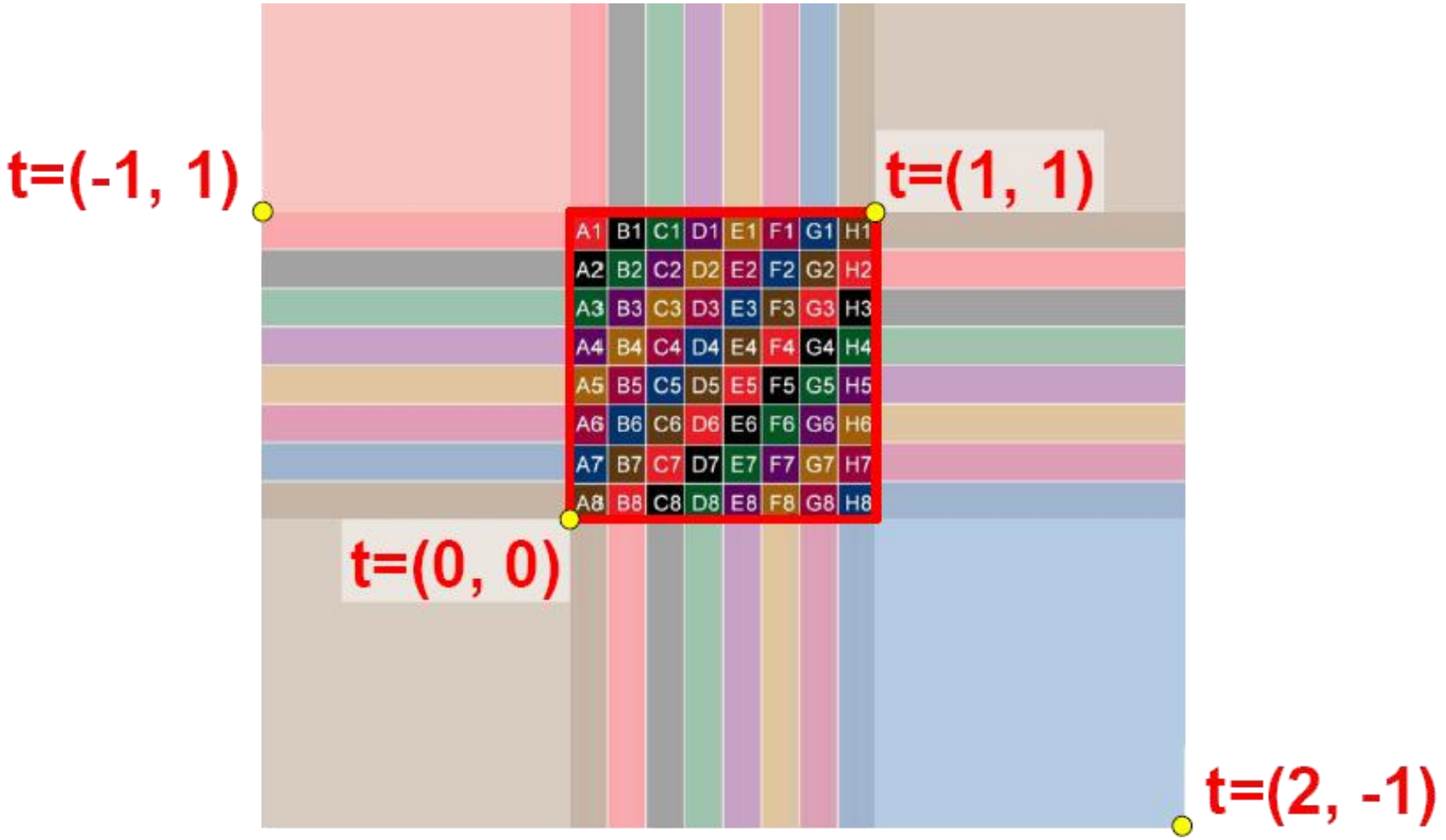
Texture Coordinates

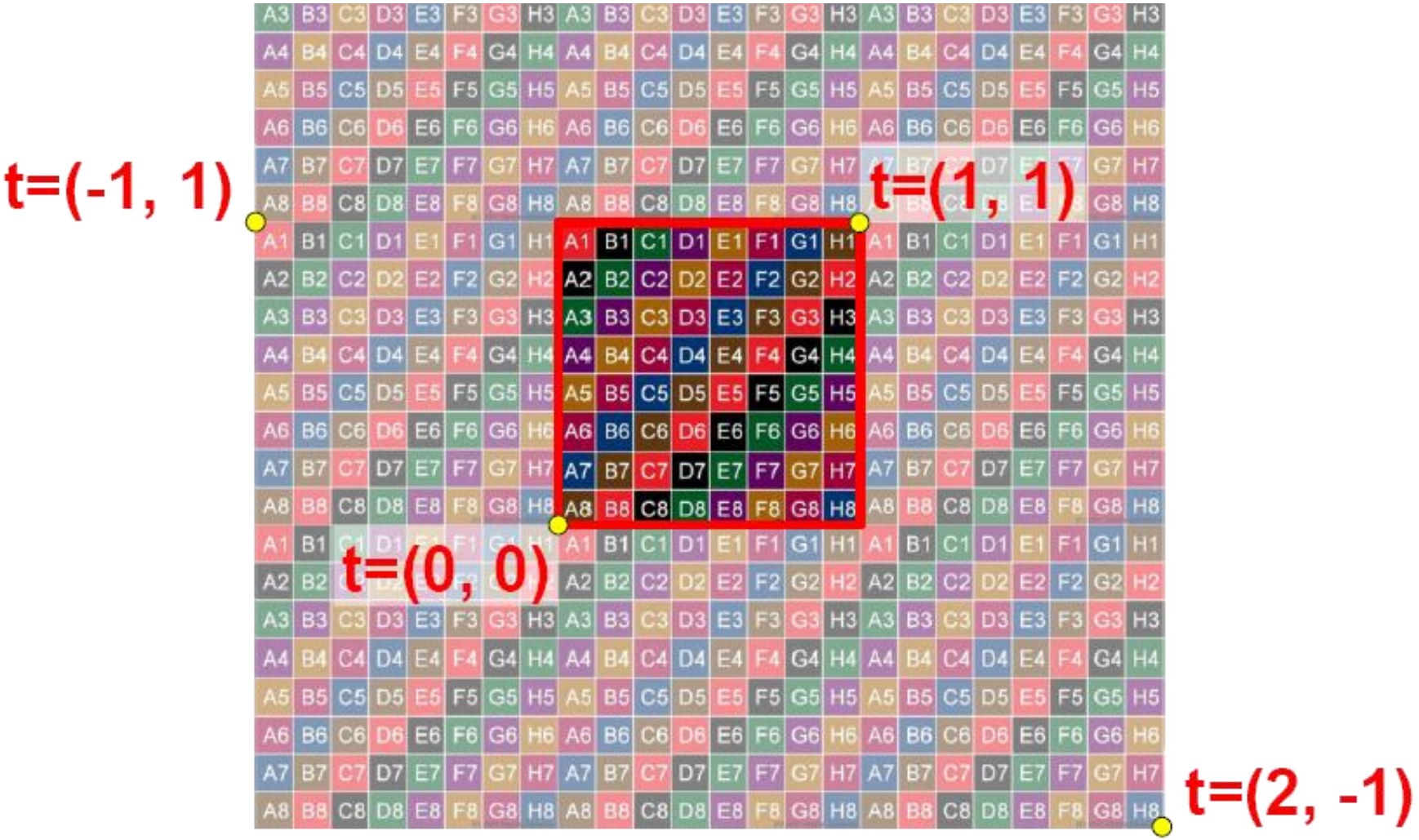
- Texture map has coordinates (0,0) to (1,1)



Texture Parameters

- GL_TEXTURE_WRAP_S, T
  - Controls what happens when you specify texture coordinates beyond [0,1].
  - GL_CLAMP
    - Value beyond standard square is equal to boundary
  - GL_REPEAT
    - Value beyond standard square is as if square is repeating (tiling)


t=(-1, 1)
t=(1, 1)
A1 B1 C1 D1 E1 F1 G1 H1
A2 B2 C2 D2 E2 F2 G2 H2
A3 B3 C3 D3 E3 F3 G3 H3
A4 B4 C4 D4 E4 F4 G4 H4
A5 B5 C5 D5 E5 F5 G5 H5
A6 B6 C6 D6 E6 F6 G6 H6
A7 B7 C7 D7 E7 F7 G7 H7
A8 B8 C8 D8 E8 F8 G8 H8
t=(0, 0)
t=(2, -1)


t=(-1, 1)
t=(1, 1)
t=(0, 0)
t=(2, -1)