

# 02561 Computer Graphics

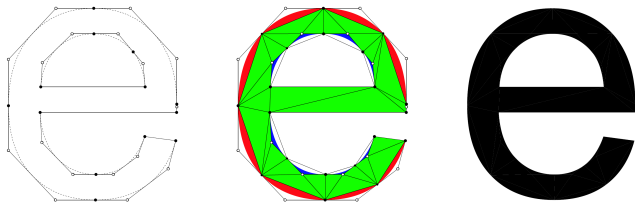
Benefits of the browser environment

Jeppe Revall Frisvad

September 2020

## How to render text properly?

- ▶ Reading text is critical to civilization.
- ▶ Rendering text properly is important.
- ▶ Scaling an image of a letter is suboptimal: (up) s s **S S** S s (down)
- ▶ Text should be rendered in vector graphics. Quality is then resolution independent.
- ▶ Characters (glyphs) are usually specified by closed paths of parametric curves.



- ▶ Browsers are good at text rendering. It's an important part of their job.
- ▶ Let's see if we can exploit this.

### Reference

- Loop, C., and Blinn, J. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 2005)* 24(3), pp. 1000–1009. July 2005.

## Rendering 2D text in WebGL

- ▶ **Head Up Display (HUD)**
  - a transparent display originally developed for aircraft.
  - ▶ Present data without requiring users to look away from their usual viewpoints.
  - ▶ Overlay textual information on a 3D scene.
- ▶ **How to make a HUD in WebGL:**
  - ▶ Create a 2D canvas element that can exploit the browser's text rendering capabilities.
  - ▶ Place the 2D canvas on top of the WebGL canvas. HTML example:

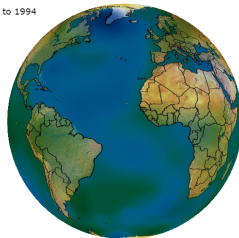
```
<canvas id="webgl" width="960" height="540" style="position: absolute; z-index: 0">No HTML5 canvas.</canvas>  
<canvas id="hud" width="960" height="540" style="position: absolute; z-index: 1"></canvas>
```
  - ▶ Writing in the HUD in Javascript:

```
function init() {
    var hud = document.getElementById('hud');
    var ctx = hud.getContext('2d');
    :
    :
}

function draw2D(ctx, year) {
    ctx.clearRect(0, 0, 960, 540);
    ctx.font = '18px "Verdana"'; ctx.fillStyle = 'rgba(0, 0, 0, 1)';
    ctx.fillText('years ' + year.toString() + ' to ' + (year + 4).toString(), 120, 50);
}
```



The pilot's view through a Rockwell Collins HGS head-up display.

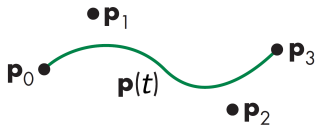


## Parametric curves

- ▶ A parametric curve is a vector valued function of a single variable.
- ▶ Example: rational  $n$ th degree power basis curve

$$C(t) = \mathbf{C}t, \quad \mathbf{C} = \begin{bmatrix} x_0 & \cdots & x_n \\ y_0 & \cdots & y_n \\ w_0 & \cdots & w_n \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} 1 & t & \cdots & t^n \end{bmatrix}^T, \quad t \in [0, 1].$$

- ▶ The vector  $C(t) = \begin{bmatrix} x & y & w \end{bmatrix}^T$  is in **homogeneous coordinates**. This means that the position in the plane is  $\mathbf{p} = (\frac{x}{w}, \frac{y}{w})$ .
- ▶ The idea of this construction is to enable matrix representation of rational curves.



## Bézier curves

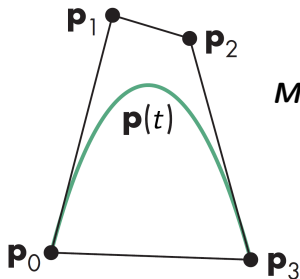
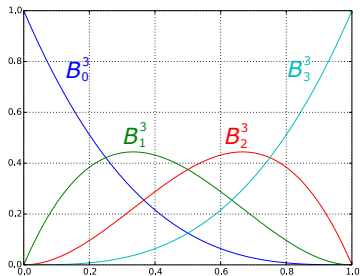
- ▶ Bernstein basis functions:  $B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$ .

- ▶ Bézier curve:

$$B(t) = \mathbf{B} \begin{bmatrix} B_0^n(t) & B_1^n(t) & \cdots & B_n^n(t) \end{bmatrix}^T, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \cdots & \mathbf{b}_n \end{bmatrix},$$

where  $\mathbf{b}_i$  are  $n+1$  Bézier control points.

- ▶ We can find matrices so that  $\mathbf{C} = \mathbf{B}\mathbf{M}^n$ , for example  $\mathbf{M}^2 = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$ .



$$\mathbf{M}_{k+1,i+1}^n = (-1)^{i-k} \binom{n}{i} \binom{i}{k}$$

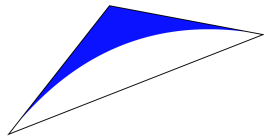
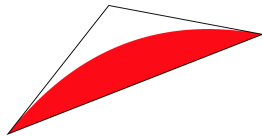
## Rational quadratic Bézier curves in triangles

- ▶ Consider the curve  $\mathbf{p}(t) = (t, t^2)$ . Or:  $F(t) = \mathbf{F}t = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}$ .

- ▶ This curve has the implicit form:  $x^2 + y = 0$ ,  
 $x, y \in [0, 1]$ .

- ▶ The Bézier control points of the curve:  $\mathbf{B} = \mathbf{F}(\mathbf{M}^2)^{-1} = \begin{bmatrix} 0 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ .

- ▶ Suppose we draw a triangle with a varying vertex attribute  $\mathbf{c} = (u, v)$  defined by the given Bézier control points:  $\mathbf{c}_0 = (0, 0)$ ,  $\mathbf{c}_1 = (0.5, 0)$ ,  $\mathbf{c}_2 = (1, 1)$ .
- ▶ For every fragment, we can now evaluate  $f(u, v) = u^2 + v$  to see if the fragment is close to the curve  $|f(u, v)| < \epsilon$  or on one side or the other.
- ▶ The triangle shape (vertex positions) determine the shape of the curve.
- ▶ The same control points work for **any** rational quadratic Bézier curve.



Reference

- Loop, C., and Blinn, J. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics (SIGGRAPH 2005)* 24(3), pp. 1000–1009. July 2005.

## Curve rendering in the HUD

- ▶ The head up display (HUD) provides vector graphics rendering of text and curves.
- ▶ It has a clear function: `ctx.clearRect(x, y, width, height)`.
- ▶ And a number of draw options for setting state:
  - ▶ `ctx.font` (a text string describing the font)
  - ▶ `ctx.textAlign` (alignment: left, right, center)
  - ▶ `ctx.fillStyle` (a color or pattern)
  - ▶ `ctx.strokeStyle` (a color or pattern)
  - ▶ `ctx.lineWidth` (in pixel)
- ▶ Text is drawn using: `ctx.fillText(text, x, y)`.
- ▶ The coordinates  $(x, y)$  are in canvas space.
- ▶ Bézier curves are drawn in a similar way:  
`ctx.moveTo( $a_x$ ,  $a_y$ ); ctx.quadraticCurveTo( $b_x$ ,  $b_y$ ,  $c_x$ ,  $c_y$ ); ctx.stroke();`  
`ctx.moveTo( $a_x$ ,  $a_y$ ); ctx.bezierCurveTo( $b_x$ ,  $b_y$ ,  $c_x$ ,  $c_y$ ,  $d_x$ ,  $d_y$ ); ctx.stroke();`  
where ***a***, ***b***, ***c***, ***d*** are the control points in canvas space coordinates.



# Rendering vector graphics in a 3D scene

- ▶ To render text well in 3D, we need Loop's and Blinn's method or its newest reincarnation.



- ▶ A simplistic alternative is to use communication between 2D canvas and WebGL.
- ▶ What we draw in a canvas, we can store in GPU memory as a texture image.

## References

- Loop, C., and Blinn, J. Rendering vector art on the GPU. In *GPU Gems 3*, Chapter 25, pp. 543–562. Pearson Education, 2008.
- Dokter, M., Hladky, J., Parger, M., Schmalstieg, D., Seidel, H.-P., and Steinberger, M. Hierarchical rasterization of curved primitives for vector graphics rendering on the GPU. *Computer Graphics Forum* 38(2):93–103. May 2019.



# Canvas as texture

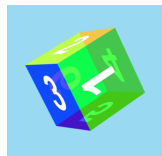
- ▶ WebGL supports direct use of a canvas element as texture image data.

```
var tex = gl.createTexture();  
gl.bindTexture(gl.TEXTURE_2D, tex);  
gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);  
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);  
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);  
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);  
draw2D(ctx);  
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, ctx.canvas);
```

- ▶ We can map what was drawn in the drawing program to a surface in a 3D scene.
- ▶ We can draw font characters in appropriate size to a hidden canvas and use these to get decent text in 3D:

```
<canvas id="text" style="position: absolute; visibility: hidden"></canvas>
```

- ▶ A rolling semi-transparent die with numbers as an example



## Adapting to browser window size (resize)

- ▶ The browser triggers a "resize" event when the user resizes the browser window.
- ▶ We can use this to ensure that a canvas always fills out the entire client area.
- ▶ We can also set the font size to be used in a HUD or a hidden text canvas.
- ▶ Example from the init function of the rolling die program:

```
function resize() {  
    canvas.width = window.innerWidth;  
    canvas.height = window.innerHeight;  
    gl.viewport(0, 0, canvas.width, canvas.height);  
    var fontsize = Math.ceil(canvas.height*0.1);  
    makeTextTexture(gl, textctx, texttex, '0123456789', fontsize + 'px ' + font, ...);  
}  
window.addEventListener('resize', resize);  
resize();
```

- ▶ This is important when making a program suitable for different devices.



## Sound effects

- ▶ The browser also makes it easy to include sound effects.
- ▶ We can make a function that becomes a playable sound object:

```
function sound(src) {  
  this.sound = document.createElement("audio");  
  this.sound.src = src;  
  this.sound.setAttribute("preload", "auto");  
  this.sound.setAttribute("controls", "none");  
  this.sound.style.display = "none";  
  document.body.appendChild(this.sound);  
  this.play = function () {  
    var playPromise = this.sound.play();  
    if(playPromise !== undefined) {  
      playPromise.then(_ => { /* Show playing UI. */ }).catch(error => { /* Show paused UI. */ });  
    }  
  }  
  this.stop = function () { this.sound.pause(); }  
  this.load = function () { this.sound.load(); }  
}
```

- ▶ Initialization: `var clickSound = new sound("sounds/click.mp3");`
- ▶ and we can then later play the sound (in a click event function, for example) using `clickSound.play();`

# Time-stamped animation

- ▶ The callback function used with `requestAnimationFrame` can optionally take a timestamp argument. Example:

```
var init_anim = true;
var time0;
function tick(timestamp) {
  if(init_anim) {
    time0 = timestamp;
    if(time0) init_anim = false;
  }
  if(!init_anim) {
    var t = timestamp - time0; // Animation time
    .
    .
    .
  }
  requestAnimationFrame(tick);
}
```



- ▶ Time  $t$  is measured in milliseconds.
- ▶ If `time0` is reset for every frame, we can do relative updates based on the time since last frame  $t$  and the current velocity of the object.
- ▶ If `time0` is only reset at animation initialization, we can do keyframe animation: Pre-staged interpolation between keyframes based on the time  $t$  since initialization.

## Other graphics techniques needed to make a rolling die



- ▶ Perspective projection and object rotation.
  - ▶ We use the concept of homogeneous coordinates in 3D and construct  $4 \times 4$  transformation matrices for manipulating scene content.
- ▶ Blending and draw order.
  - ▶ We use the alpha-channel (in RGBA colors) and specify a blend function.
  - ▶ Drawing order becomes important when surfaces are semi-transparent.
- ▶ Texturing.
  - ▶ Mapping images to surface.
  - ▶ Using filters to deal with aliasing problems (pixelation / staircases / jaggies).
  - ▶ Drawing to an off-screen buffer and using it as a texture.
- ▶ These are all subjects that we will investigate during the course.
- ▶ The rolling die program has been uploaded to DTU Inside for you to explore. Think of it as a simple demo of many graphics concepts.

## Drawing multiple objects

- ▶ Vertex specification best practices:
  - ▶ As few vertex buffers as possible (separate buffers for different attributes is ok).
  - ▶ Switch vertex buffer, shader program, etc. as infrequently as possible.
  - ▶ Standard mode of operation: one draw call per object.
  - ▶ Include multiple objects in one draw call when possible (batch drawing).
  - ▶ Use the same vertices for drawing several different objects by changing color/transformation variables (instancing).

[https://www.khronos.org/opengl/wiki/Vertex\\_Specification\\_Best\\_Practices](https://www.khronos.org/opengl/wiki/Vertex_Specification_Best_Practices)