Wprowadzenie do grafiki maszynowej. Shadery Geometrii

Aleksander Denisiuk
Uniwersytet Warmińsko-Mazurski
Olsztyn, ul. Słoneczna 54
denisjuk@matman.uwm.edu.pl

Shadery Geometrii

Shader geometrii

Zastosowanie: krzywe Béziera

Najnowsza wersja tego dokumentu dostępna jest pod adresem

http://wmii.uwm.edu.pl/~denisjuk/uwm

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

Shader geometrii

Wstęp

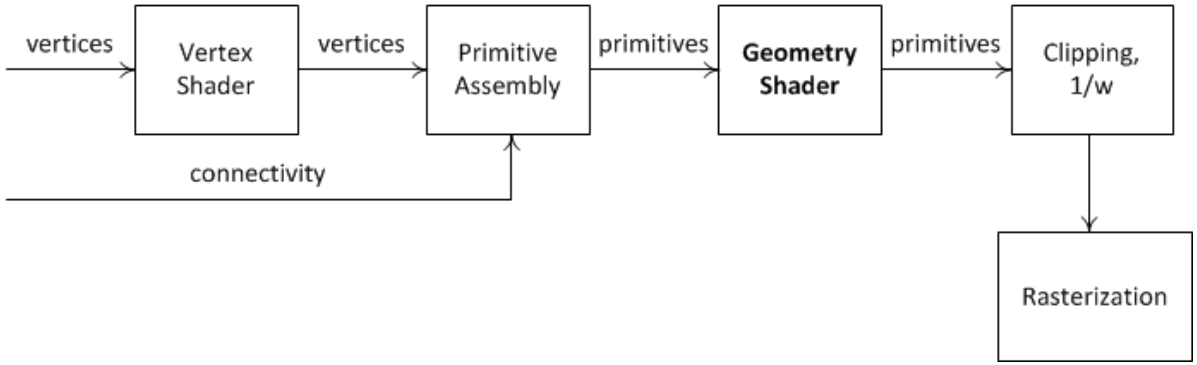
Shader geometrii
❖ Wstęp
❖ Nowe prymitywy graficzne
❖ Wejście/Wyjście
❖ Podsumowanie
Zastosowanie: krzywe Béziera

- Między shaderem wierzchołków a shaderem fragmentów
- Każdy prymityw graficzny
- Na wejściu prymityw graficzny (`points`, `lines`, etc)
- Na wyjściu nowy prymityw graficzny

Potok renderingu

Shader geometrii
- ❖ Wstęp
- ❖ Nowe prymitywy graficzne
- ❖ Wejście/Wyjście
- ❖ Podsumowanie

Zastosowanie: krzywe Béziera



Lines/Line Strip with adjecency

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

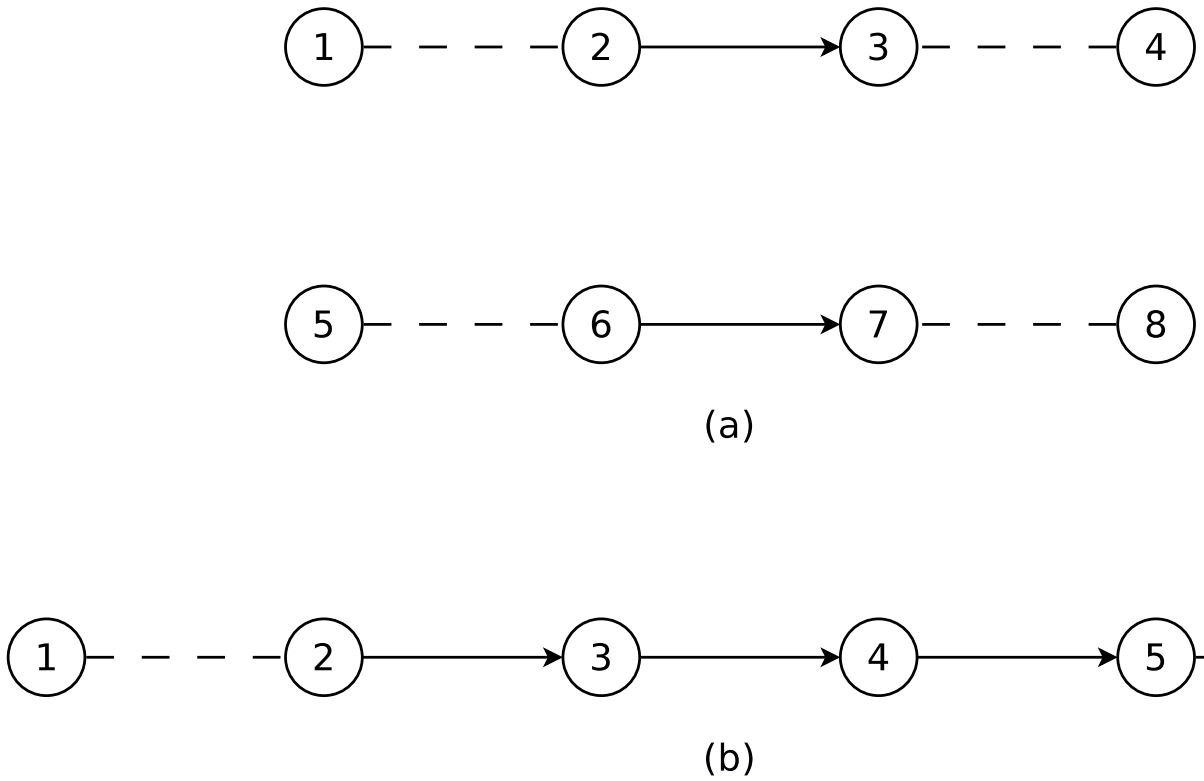
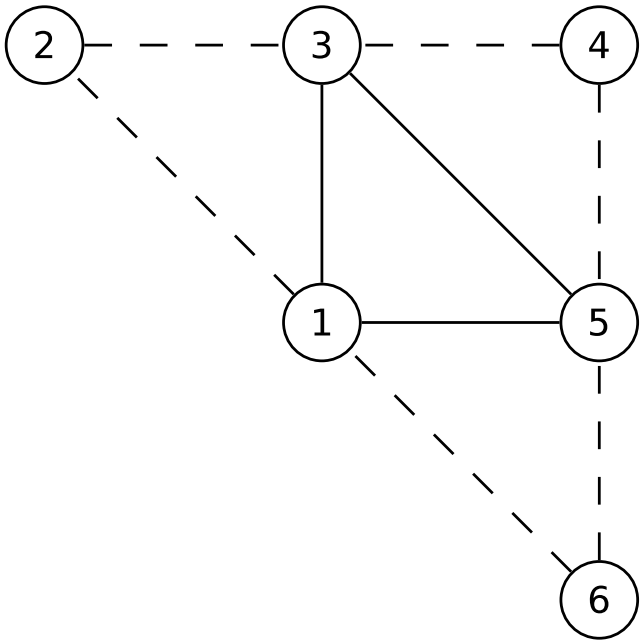


Figure 2.4. Lines with adjacency (a) and line strips with adjacency (b). The vertices connected with solid lines belong to the main primitives; the vertices connected by dashed lines are the adjacent vertices that may be used in a geometry shader.

Triangles with adjacency

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera



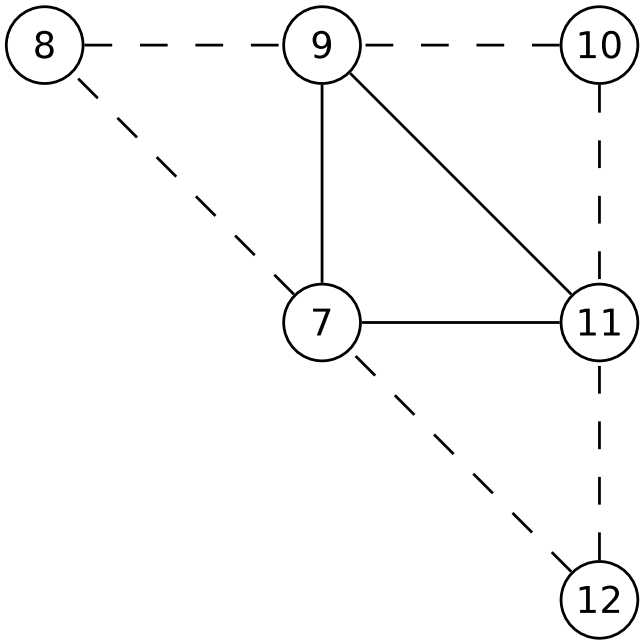


Figure 2.5. Triangles with adjacency. The vertices connected with solid lines belong to the main primitive; the vertices connected by dashed lines are the adjacent vertices that may be used in a geometry shader.

Triangle strip with adjacency

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

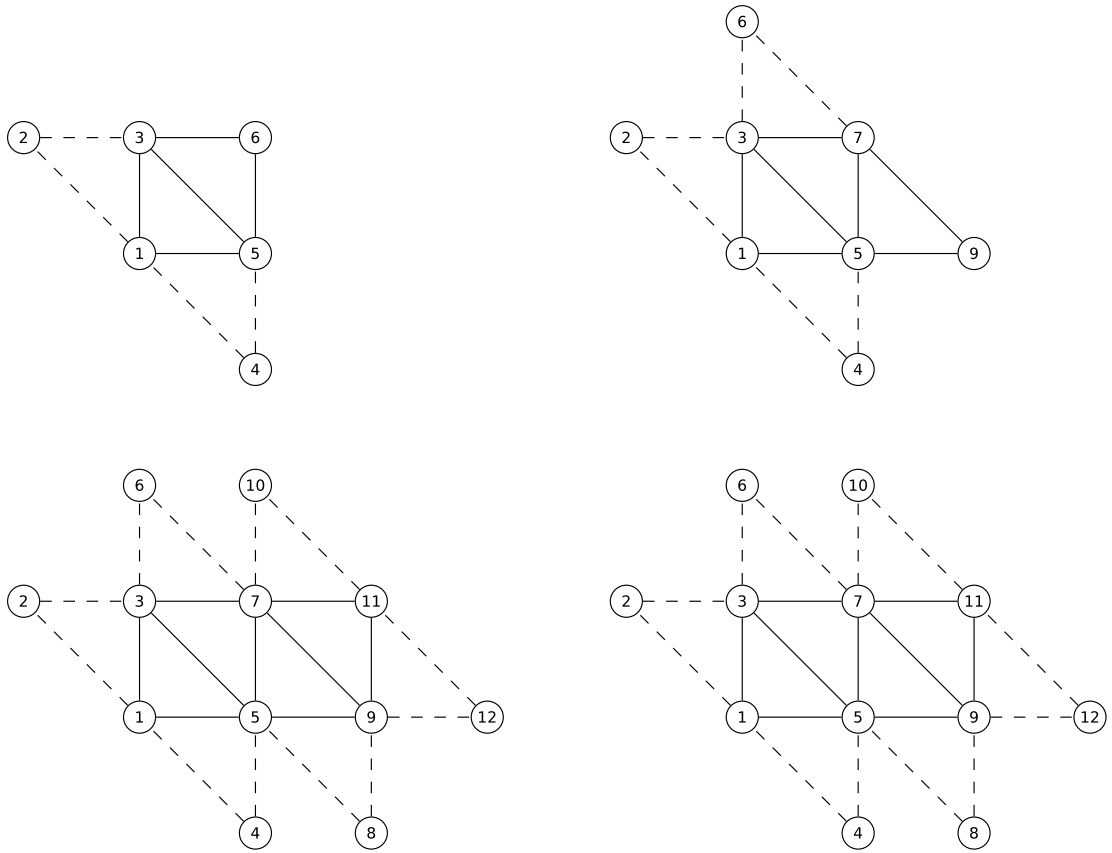


Figure 2.6. Triangle strips with adjacency. The vertices connected with solid lines belong to the main primitives; the vertices connected by dashed lines are the adjacent vertices that may be used in a geometry shader.

Wejście i tryb

Shader geometrii
❖ Wstęp
❖ Nowe prymitywy graficzne
❖ Wejście/Wyjście
❖ Podsumowanie
Zastosowanie: krzywe Béziera

- `points` $\Leftarrow$ `GL_POINTS`
- `lines` $\Leftarrow$ `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`
- `triangles` $\Leftarrow$ `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`
- `lines_adjacency` $\Leftarrow$ `GL_LINES_ADJACENCY`, `GL_LINE_STRIP_ADJACENCY`
- `triangles_adjacency` $\Leftarrow$ `GL_TRIANGLES_ADJACENCY`, `GL_TRIANGLE_STRIP_ADJACENCY`
- przykład:

```
layout(triangles) in;
```

Wyjście

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

- `points`
- `line_strip`
- `triangle_strip`
- przykład:

```
layout(triangle_strip) out;
```

Maksymalna ilość wyemitowanych wierzchołków

Shader geometrii
❖ Wstęp
❖ Nowe prymitywy graficzne
❖ Wejście/Wyjście
❖ Podsumowanie
Zastosowanie: krzywe Béziera

```
layout(max_vertices=4) out;
layout(triangle_strip, max_vertices=4) out;
```

- ograniczenie (nie mniej niż 256), przykładowo 1024

```
glGetIntegerv(GL_MAX_GEOMETRY_OUTPUT_VERTICES)
```

Ilość wywołań

Shader geometrii
❖ Wstęp
❖ Nowe prymitywy graficzne
❖ Wejście/Wyjście
❖ Podsumowanie
Zastosowanie: krzywe Béziera

```
layout(invocations = 5) in;
```

- od OpenGL 4.0
- `gl_InvocationID`

Zmienne wejściowe

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

```
gl_in[]
gl_PrimitiveIDIn
gl_InvocationID
```

Tablica `gl_in[]`

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

```
gl_Position
gl_PointSize
gl_ClipDistance []
```

Przekazywanie informacji z shadera wierzchołków

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

- w shaderze wierzchołków

```
out vec3 normal;
```

- w shaderze geometrii

```
in vec3 normal[];
```

Zmienne uniform

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

- shader geometrii ma dostęp do zmiennych uniform

Wynik

Shader geometrii
- ❖ Wstęp
- ❖ Nowe prymitywy graficzne
- ❖ Wejście/Wyjście
- ❖ Podsumowanie

Zastosowanie: krzywe Béziera

- `gl_Position`
- `gl_PointSize`
- `gl_ClipDistance[]`
- `gl_PrimitiveID`
- `gl_Layer`
- `gl_ViewportIndex`

Wyjściowe zmienne

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

```
out      vec3 vertexNormal;
flat out vec3 faceNormal;
```

- `flat` nie interpoluje się

  ```
  glProvokingVertex(GL_FIRST_VERTEX_CONVENTION)
  glProvokingVertex(GL_LAST_VERTEX_CONVENTION)
  ```

- po każdym `EmitVertex()` wartości zmiennych są nieokreślone

Podsumowanie

Shader geometrii
- Wstęp
- Nowe prymitywy graficzne
- Wejście/Wyjście
- Podsumowanie

Zastosowanie: krzywe Béziera

- za pomocą shadera geometrii można:
  - filtrować prymitywy
  - zmieniać typ prymitywu
  - rozdzielać prymitywy po warstwach tekstury sześcianowej (tworzyć kopie warstw)
  - obliczać parametry, dotyczące całego prymitywa (wektor normalny do ściany)
  - dla prymitywów z przyleganiem można obliczyć dla wierzchołków wektory styczne i normalne
  - utworzyć nową geometrię, nie jest zalecane tworzenie dużej ilości nowych wierzchołków
    - dla generacji dużych ilości nowej geometrii używa się shaderów tesselacji (*tesselation control shader*, *tesselation evaluation shader*), dostępnych od OpenGL 4.0

Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- `Bezier`
- `Program`
- `Window`
- Splajn Catmulla-Roma

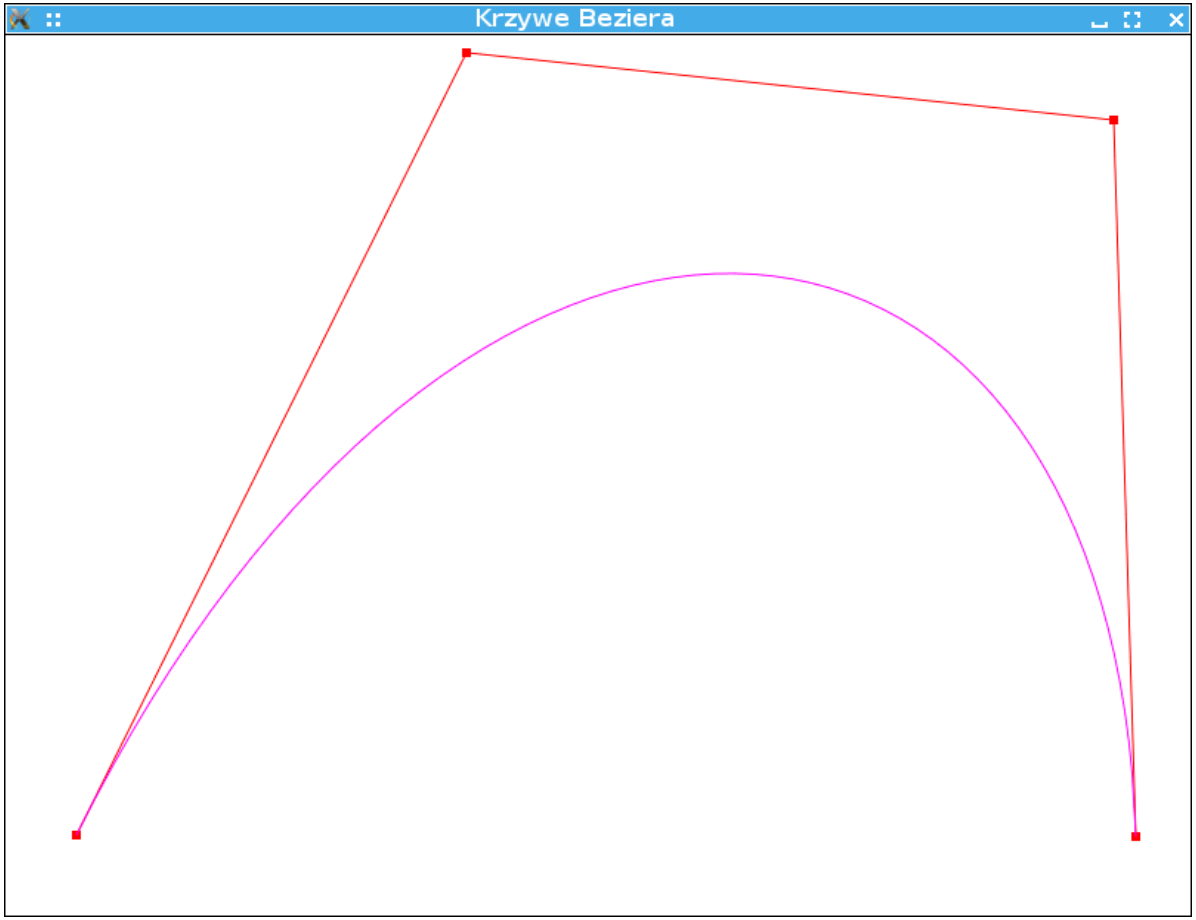
Zastosowanie: krzywe Béziera

Przykład zastosowania: krzywe Béziera

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- `Bezier`
- `Program`
- `Window`
- Splajn Catmulla-Roma



Program

Shader geometrii
Zastosowanie: krzywe Béziera
❖ Przykład
❖ Program
❖ Shadery
❖ Algorytm de Casteljau
❖ Bezier
❖ Program
❖ Window
❖ Splajn Catmulla-Roma

- Wyświetla poszczególne punkty, zaznaczone myszką
- Łączy je łamaną
- Jak są już cztery punkty, wyświetla krzywą Béziera

Shader Wierzchołków

Shader geometrii
Zastosowanie: krzywe Béziera
❖ Przykład
❖ Program
❖ Shadery
❖ Algorytm de Casteljau
❖ Bezier
❖ Program
❖ Window
❖ Splajn Catmulla-Roma

```
#version 430

layout(location=0) in vec2 in_position;

out vec4 color;

void main(void){
  gl_Position = vec4(in_position,0,1);
  color = vec4(1,0,0,1);
}
```

Shader Fragmentów

Shader geometrii
Zastosowanie: krzywe Béziera
❖ Przykład
❖ Program
❖ Shadery
❖ Algorytm de Casteljau
❖ Bezier
❖ Program
❖ Window
❖ Splajn Catmulla-Roma

```
in vec4 color;
out vec4 out_color;

void main(void){
    out_color = color;
}
```

Shader Geometrii

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
#version 430

layout (lines_adjacency) in;
layout (line_strip, max_vertices=65) out;

out vec4 color;

void main(void){
  int i;
  color=vec4(1,0,1,1);

  gl_Position = gl_in[0].gl_Position;
  EmitVertex();

 ..............................
```

Algorytm de Casteljau

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

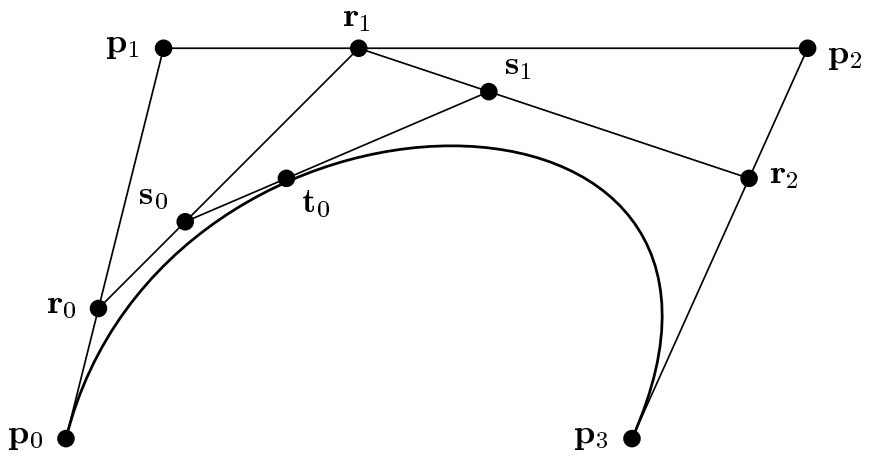


Figure VII.4: The de Casteljau method for computing $\mathbf{q}(u)$ for $\mathbf{q}$ a degree three Bézier curve. This illustrates the $u = 1/3$ case.

Algorytm de Casteljau. Implementacja

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
vec4 r0, r1, r2, s0, s1;
float u=0.0, h=1.0/64.0;
for(i=1; i<64; i++){ u +=h;
 r0=mix(gl_in[0].gl_Position,
        gl_in[1].gl_Position,u);
 r1=mix(gl_in[1].gl_Position,
        gl_in[2].gl_Position,u);
 r2=mix(gl_in[2].gl_Position,
        gl_in[3].gl_Position,u);
 s0=mix(r0, r1, u); s1=mix(r1, r2, u);
 gl_Position = mix(s0, s1, u);
 EmitVertex();
 }
 gl_Position = gl_in[3].gl_Position;
 EmitVertex();
 EndPrimitive();
}
```

Klasa Bezier

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
class Bezier{
 public:
   void Initialize();
   void Draw(const BezierProgram & );
   ~Bezier();
   Bezier(void);
   void AddPoint(GLfloat*v);

 private:
   GLuint max_points_;
   GLuint num_points_;
   GLuint vertices_;

   GLuint vao_;
   GLuint vertex_buffer_;
};
```

Konstruktor

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
Bezier::Bezier(void){
    max_points_=4;
    num_points_=0;
}
```

Inicjalizacja

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
glGenVertexArrays(1, &vao_);
glBindVertexArray(vao_);

glGenBuffers(1, &vertex_buffer_);
glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer_);
glBufferData(GL_ARRAY_BUFFER,
   max_points_*2*sizeof(GLfloat),
   NULL, GL_DYNAMIC_DRAW);
glVertexAttribPointer(0, 2, GL_FLOAT,
         GL_FALSE, 0, (GLvoid*)0);
glEnableVertexAttribArray(0);

glBindVertexArray(0);
```

Dodanie punktu

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
void Bezier::AddPoint( GLfloat*v ){
 if(num_points_ < max_points_){
  glBindVertexArray(vao_);
  glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer_);
  glBufferSubData(GL_ARRAY_BUFFER,
      (num_points_++)*2*sizeof(GLfloat),
      2*sizeof(GLfloat), v);
  glBindVertexArray(0);
 }
}
```

Wyświetlenie. Linie i punkty

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
void Bezier::Draw(const BezierProgram &program){
    glBindVertexArray(vao_);
    glUseProgram( program.GetLineProgram() );
    glPointSize(8);
    glDrawArrays(GL_POINTS, 0, num_points_);
    glPointSize(1);
    glDrawArrays(GL_LINE_STRIP, 0, num_points_);

 ..............................
```

Rozmiar punktu

Shader geometrii
Zastosowanie: krzywe Béziera
❖ Przykład
❖ Program
❖ Shadery
❖ Algorytm de Casteljau
❖ Bezier
❖ Program
❖ Window
❖ Splajn Catmulla-Roma

- W programie

```
glPointSize(n);
```

- Alternatywnie: w shaderze

```
gl_PointSize = n;
```

  - ✦ powinien być aktywowany tryb

```
glEnable(GL_PROGRAM_POINT_SIZE);
```

Wyświetlenie. Krzywa

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
.................................
 if(num_points_ == max_points_){
   glUseProgram( program.GetBezierProgram() );
   glDrawArrays(GL_LINES_ADJACENCY, 0,
                                    num_points_);
 }
 glBindVertexArray(0);
 glUseProgram(0);
}
```

Destruktor

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
Bezier::~Bezier(){
    glBindVertexArray(vao_);
    glDisableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glDeleteBuffers(1, &vertices_);
    glBindVertexArray(0);
    glDeleteVertexArrays(1, &vao_);
}
```

Klasa `BezierProgram`, dane prywatne

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma


```
class BezierProgram{
private:
  GLuint line_program_;
  GLuint bezier_program_;

  GLuint vertex_shader_;
  GLuint fragment_shader_;
  GLuint geometry_shader_;

  GLuint LoadAndCompileShaderOrDie(
    const char* source_file, GLenum type);
  GLuint LinkProgramOrDie(
    GLint vertex_shader,
    GLint fragment_shader,
    GLint geometry_shader=0);

.............................
```

Klasa `BezierProgram`, dane publiczne

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma


```
 ..............................
public:
    void Initialize(const char *vertex,
        const char *fragment,
        const char *geometry)
    ~BezierProgram();

    GLuint GetLineProgram() const{
            return line_program_;
           }
    GLuint GetBezierProgram() const{
            return bezier_program_;
           }
};
```

Inicjalizacja

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
void Program::Initialize(const char *vertex,
const char *fragment, const char *geometry){
  initializeOpenGLFunctions();
  vertex_shader_
    = LoadAndCompileShaderOrDie(vertex,
                  GL_VERTEX_SHADER);
  fragment_shader_
    = LoadAndCompileShaderOrDie(fragment,
                  GL_FRAGMENT_SHADER);
  geometry_shader_
    = LoadAndCompileShaderOrDie(geometry,
                  GL_GEOMETRY_SHADER);
  program_ = LinkProgramOrDie(vertex_shader_,
                  fragment_shader_);
  bezier_program_ = LinkProgramOrDie(
    vertex_shader_, fragment_shader_,
    geometry_shader_);
}
```

Linkowanie

Shader geometrii
Zastosowanie: krzywe Béziera
- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

```
GLuint BezierProgram::LinkProgramOrDie(
 GLint vertex_shader,
 GLint fragment_shader,
 GLint geometry_shader){
  GLuint new_program = glCreateProgram();
  glAttachShader(new_program, vertex_shader);
  glAttachShader(new_program, fragment_shader);
  if (geometry_shader>0)
      glAttachShader(new_program,
                     geometry_shader);
  glLinkProgram(new_program);

 ..............................
```

Destruktor. Odłączenie shaderów

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma


```
glUseProgram(0);
 glDetachShader(line_program_, vertex_shader_);
 glDetachShader(line_program_,
                                fragment_shader_);

 glDetachShader(bezier_program_,
                                vertex_shader_);
 glDetachShader(bezier_program_,
                                fragment_shader_);
 glDetachShader(bezier_program_,
                                geometry_shader_);

 ..............................
```

Destruktor. Usuwanie programów

Shader geometrii
Zastosowanie: krzywe Béziera
❖ Przykład
❖ Program
❖ Shadery
❖ Algorytm de Casteljau
❖ Bezier
❖ Program
❖ Window
❖ Splajn Catmulla-Roma

```
.....................................
  glDeleteShader(geometry_shader_);
  glDeleteShader(fragment_shader_);
  glDeleteShader(vertex_shader_);

  glDeleteProgram(line_program_);
  glDeleteProgram(bezier_program_);
}
```

Klasa `Window`. Kliknięcie

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- `Bezier`
- `Program`
- `Window`
- Splajn Catmulla-Roma


```
void Window::MouseButton(int button,
      int action, int /*mods*/) {
  if (button == GLFW_MOUSE_BUTTON_LEFT) {
    if (action == GLFW_RELEASE) {
      GLfloat x_y_pos[2];
      double x_pos, y_pos;
      glfwGetCursorPos(*this, &x_pos, &y_pos);
      x_y_pos[0]
          = (float)x_pos*2.0f/width_ - 1.0f;
      x_y_pos[1]
          =  -(float)y_pos*2.0f/height_ + 1.0f;
      bezier_.AddPoint(x_y_pos);
    }
  }
}
```

Splajn Catmulla-Roma

Shader geometrii

Zastosowanie: krzywe Béziera

- Przykład
- Program
- Shadery
- Algorytm de Casteljau
- Bezier
- Program
- Window
- Splajn Catmulla-Roma

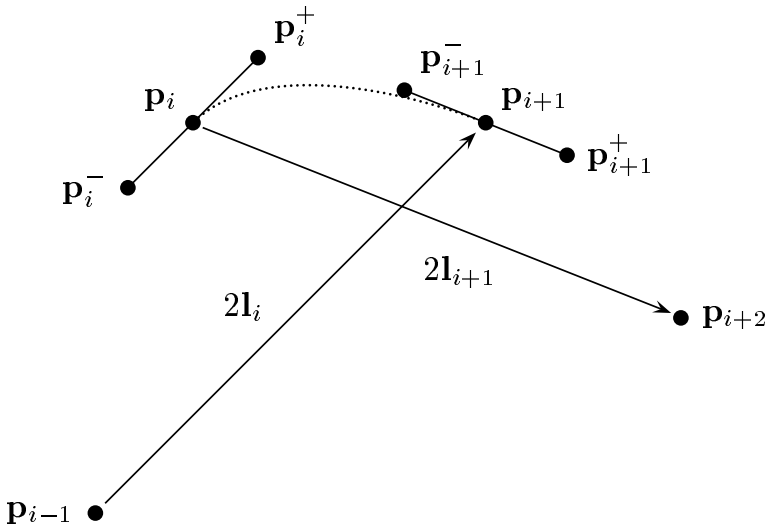


Figure VII.22: Defining the Catmull-Rom spline segment from the point $\mathbf{p}_i$ to the point $\mathbf{p}_{i+1}$. The points $\mathbf{p}_i^-$, $\mathbf{p}_i$, and $\mathbf{p}_i^+$ are collinear and parallel to $\mathbf{p}_{i+1} - \mathbf{p}_{i-1}$. The points $\mathbf{p}_i$, $\mathbf{p}_i^+$, $\mathbf{p}_{i+1}^-$, and $\mathbf{p}_{i+1}$ form the control points of a degree three Bézier curve, which is shown as a dotted curve.

$$l_i = \frac{p_{i+1} - p_{i-1}}{2}, \quad p_i^{\pm} = p_i \pm \frac{1}{2} l_i$$