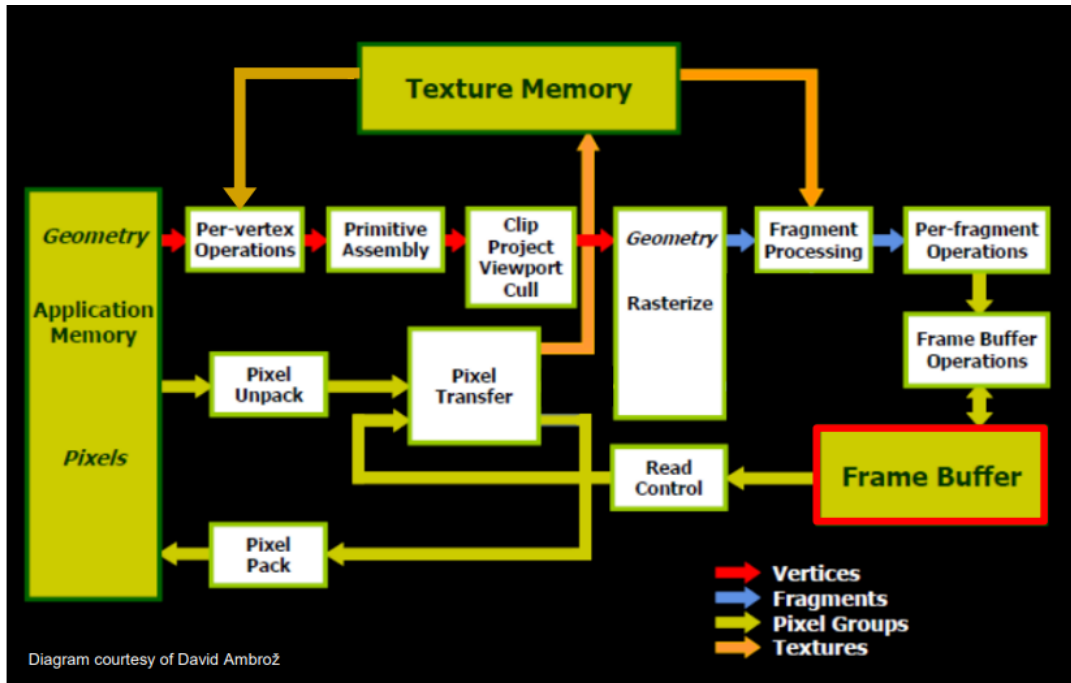


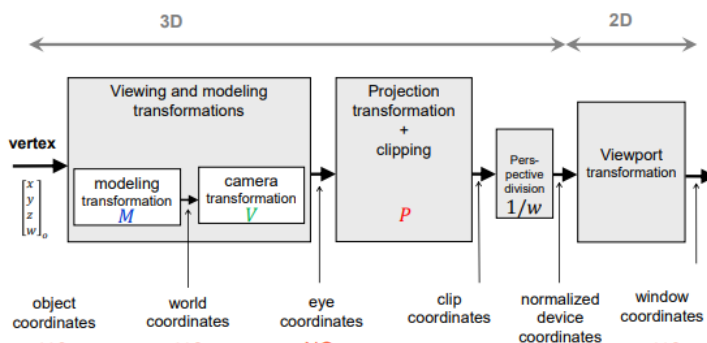
# PGR

## 1. Rastrový zobrazovací řetězec

Rastrový zobrazovací řetězec, jeho fixní a programovatelné bloky. Popis a funkce.



- **Vertex shader** - operace na každém vrcholu a jejich vlastnostech (pozice, normála, barva...), spuštěno pro každý vrchol, transformace souřadnic, normály (+normalizace), tex. souřadnic (+jejich možná generace), per-vertex lighting a další atributy
- **Primitive assembly** - sestavení primitiv z vrcholů (trojúhelníku/quadů)
- **Clipping** - primitiva ležící mimo pohledový jehlan jsou zahozena a nejdou pipeline dále, primitiva ležící na hranici pohledového jehlanu jsou rozdělena na několik primitiv, tak, aby se vešly do pohledového jehlanu
- **Back-face culling** - stěny, které směřují směrem od kamery (pomocí CW nebo CCW zadání vrcholů / normály) jsou vynechány a dále do pipeline nejdou
- **Dělení Wc** - převod z homogenních souřadnic vydělením x, y a z pomocí W clip coords = -z eye coords (normalizované souřadnice zařízení, <-1, 1> NSZ)
- **Viewport transform** - umístí vrcholy do okna z NSZ souřadnic, x a y jsou souřadnice v okně a z je hloubka využívaná pro správné pořadí při vykreslování
- **Geometry shader** - možnost volitelného geometry shaderu, lze přidat trojúhelníky navíc
- **Rasterizace** - vytvořené trojúhelníky se rasterizují a předělají na fragmenty (approximace)
- **Fragment shader** - Operace na každém fragmentu (kandidát na pixel s extra info vytvořen rasterizerem, neplatí vždy že pixel = fragment, lze více fragmentů na jeden pixel, ale vždy alespoň jeden), aplikace textur, mlha, per-pixel lighting (lepší), operace na interpolovaných hodnotách



## Perspektivně správná interpolace.




- souřadnice vrcholu a dělitel W (viz. sekce výš) jsou separátně lineárně interpolovány

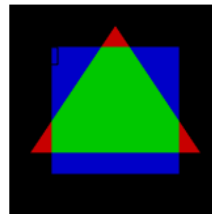
- souřadnice jsou interpolovány podle primitiva, clipped a vynásobeny lin. interp.  $1/W_c$ , kde  $W_c = -z$  eye coords
- atributy vrcholů jsou vynásobeny  $W_c$ , pak jsou lin. interp. a pak vynásobeny lin. interp.  $1/W_c$ , kde  $W_c = -z$  eye coords

## Vrstvy obrazové paměti a jejich účel (použití v bloku operací a testů).

- **Depth buffer** - ukládá data o hloubce každého pixelu jako vzdálenost od near plane, používá se pro správné překrývání objektů, hodnoty v rozmezí  $<0.0, 1.0>$  o velikosti 24 bitů;
- **Stencil buffer** - používá se pro označení jednotlivých oblastí na obrazovce, například pro obrysy, odrazy v zrcadle nebo vodě, většinou 8 bitů na pixel
- **Color front** - viditelná aktuálně zobrazená vrstva, většinou 8+8+8+8 bitů (RGBA)
- **Color back** - neviditelná vrstva do které se kreslí a je pak vyměněna s color front
- každá barevná vrstva má levou (hlavní) a pravou verzi, využívanou pro stereo obraz
- Testy:
  - Pixel ownership test - testuje zda kreslený pixel náleží v okně aplikace
  - Scissor test - t. z. k. p. leží v definovaném obdélníku výstřižku, použitelné pro update jenom části okna, rychlá verze stencil testu
  - Depth test - t. z. hloubka k. p. je menší než ta hloubka pixelu, co už tam leží a přepíše ho, řeší viditelnost pixelů, přesnost klesá s větší near to far vzdáleností, Z-fighting - dva objekty mají stejnou hloubku a jeden snímek je vidět jeden a druhý druhý -> blikání
  - Stencil test - musí projít i depth testem, pak se definuje funkce pro dané kreslení viz. příklad

Drawing to area given by the triangle

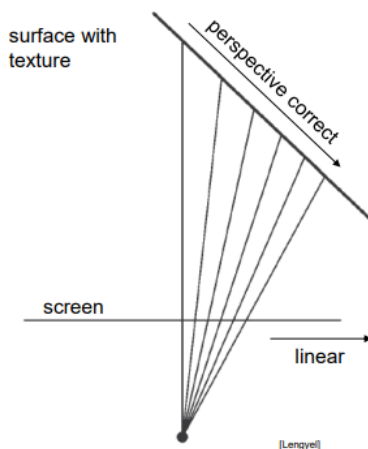
1. Draw red triangle  and fill stencil with 1
2. Draw green square  only where stencil contains 1 (only green part without corners remains)
3. Draw blue square  only where stencil does not contain 1 (only red corners remain)



- Operace:
  - Blending - míchání přichodí barvy a barvy v frame bufferu, využíváno pro průhledné objekty, nastavitelné podle čeho se to míchá (srcAlpha, destAlpha ...), u průhledných objektů je problém v pořadí vykreslování -> opaque nejdřív pak průhledné podle vzdálenosti
  - Logické operace - udávají co dělat se Source a Destination, clear, copy, invert,

## Texture - zobrazování (způsoby kombinace s osvětlením), filtrování, mip-mapping, mapa prostředí.

- vlastnost povrchu popisující drobné detaily, definuje barvu, strukturu a kvalitu
- různé texture mohou definovat různé věci - barva, odrazy světla, normály, displacement, transparency (alpha barvy)
- souřadnice [s, t, p, q] (podle dimenze, 2D nejčastěji), jeden prvek = texel
- normalized  $<0.0, 1.0>$ , non-normalized  $<0.0, m*n>$  souřadnice U, V (=s,t)
- používá se inverzní mapping - začne se v screen space -> projekce fragmentu do texture, možná interpolace texelů; texture coords (u, v) pro bod (x, y, z) z (xs, ys) screen coords
- komplexní pro nejjednodušší objekty -> před počítáno při vytváření modelu a v runtime interpolováno mezi body pomocí Perspektivně správně

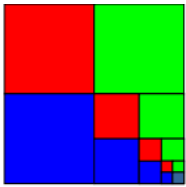


interpolace.

- texely málo kdy odpovídají pixelům a nastává problém jak pixel obarvit -> filtrování textur
- zvětšení - nejbližší texel -> aliasing, fast; lineárně interpolované 2x2 texelů -> slow ale smooth
- zmenšení - nelze nejbližší ani linear -> mip mapping

- mip-mapping - pyramida textur se zmenšující se rozlišením, kde se daná textura použije podle velikosti pixelu v texture space, možnost nearest nebo linear mezi úrovněmi

e



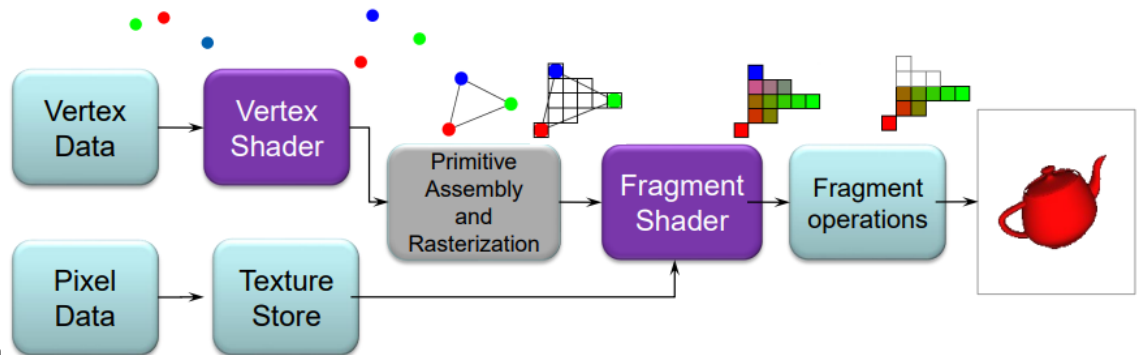
Mipmap storage – amount of memory is increased only by 1/3

- texture mají možnost pro každou osu definovat, co dělat když dojde přes interval  $<0.0, 1.0>$ , Repeat, clamp to edge, clamp to border, mirrored repeat
- Sjednocení textury a světla - osvětlený model (bílá 1 až černá 0) je vynásoben texturou
- Možnost kombinovat více textur pomocí alpha průhlednosti
- Environmental Mapping - sférická nebo kubická mapa, na objekt (koule nebo krychle) je vyrendrována celá scéna jako kdyby byl 100% reflektivní, do tohoto objektu je vložen rendrovaný objekt a jeho odlesky se vypočítají pomocí mapy prostředí, používají se hlavně cube mapy (6 obdelníků s vykresleným prostředím z každého z hlavních směrů), protože sphere mapy dovolují jen jeden viewing direction

## 2. Programování pomocí shaderů.

### Programování pomocí shaderů.

- Shader - program na GPU, definuje jednu fázi renderovací pipeline, GLSL - OpenGL Shading Language, run in parallel SIMD -> very fast



- Pipeline - viz 1. otázka

### Základní datové typy v GLSL (skalár, vektor, matice, sampler).

- stejný scope jako C++ - local, global
- basic - float, double, bool, int, uint
- vektory - 2, 3 nebo 4 komponenty, float, double, bool, int
- čtvercové matice - 2x2, 3x3, 4x4 float, double
- sampler - texture sampler, lze z něho získat texel podle souřadnic (uv)

### Rozdíl mezi proměnnými in a uniform.

- uniform - globální proměnné pro celý pipeline, stejná hodnota v různých částech pipeline, čas, textura atd.
- built-in-variables - různé hodnoty v průběhu pipeline, mezi vertex a fragment shaderem interpolováno podél primitiva, poloha, barva, normála
- Atributy - informace do VS
- Inner variables - výstup jednoho shaderu a vstup druhého (VS -> rasterizer -> FS)
- Fragment data - výstup z FS, zapsán do bufferů

### Jak se přiřazují hodnoty proměnných z OpenGL do GLSL.

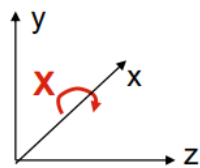
- Uniform - získání pozice (glGetUniformLocation), nastavení proměnné (glUniform\*dat. typ\*)
- Atribut - pomocí VAO, glGen..., glBind..., pak pro každý atribut glBindBuffer s hodnotami, pak glEnableVertexAttribArray(pozice\_atrib), pozice se získá pomocí glGetAttribLocation, pak glVertexAttribPointer kde se specifikuje formát atributu
- Formát atributu - pozice, počet, typ, stride - offset kde najdu další stejný atribut, offset prvního prvku od začátku pole

## 3. Typické transformace a jejich reprezentace

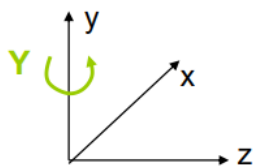
## Lineární a afinní transformace a jejich maticová reprezentace, homogenní souřadnice. Sestavení matice rotace podle jedné souřadné osy, škálování, nastavení kamery (lookAt) a záběru (viewport).

- Scale - lineární transformace, na diagonále transformační matice o kolik se každá osa natáhne (kromě 4. položky), symetrický a asymetrický

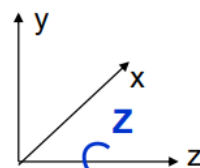
$$c = \cos \theta, s = \sin \theta$$



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$$



$$\begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}$$



$$\begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Rotation - lineární transformace
- Translation - afinní transformace, x, y, z a 1 ve 4. sloupci transformační matice
- pořadí SRT
- sheer? - (1 p) 2D (0 1)
- homogenní souřadnice - k vrcholům přidáme 4. souřadnici W (!=0 a většinou 1), platí  $(x, y, z) \rightarrow (x*w, y*w, z*w, w)$ , proč? -> afinní transformace a projekce lze v jedné matici
- lookAt matice E - umístí kameru do scény a natočí ji na nějaký bod, vstup je pozice kamery Eye, bod kam koukáme Center a směr nahoru UP,

$$\mathbf{z} = \text{normalize}(\mathbf{e} - \mathbf{c}), \quad \mathbf{upn} = \text{normalize}(\overrightarrow{\mathbf{up}})$$

$$\mathbf{x} = \text{normalize}(\mathbf{upn} \times \mathbf{z})$$

$$\mathbf{y} = (\mathbf{z} \times \mathbf{x})$$

$$E = \begin{bmatrix} x_1 & y_1 & z_1 & e_1 \\ x_2 & y_2 & z_2 & e_2 \\ x_3 & y_3 & z_3 & e_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$E^{-1}$  = Viewing matrix

- viewport - NDC  $\langle -1, 1 \rangle$  do window coordinates  $\langle \text{width}, \text{height} \rangle$ , viewport je okno aplikace, O  $(x, y)$  je pozice okna na obrazovce

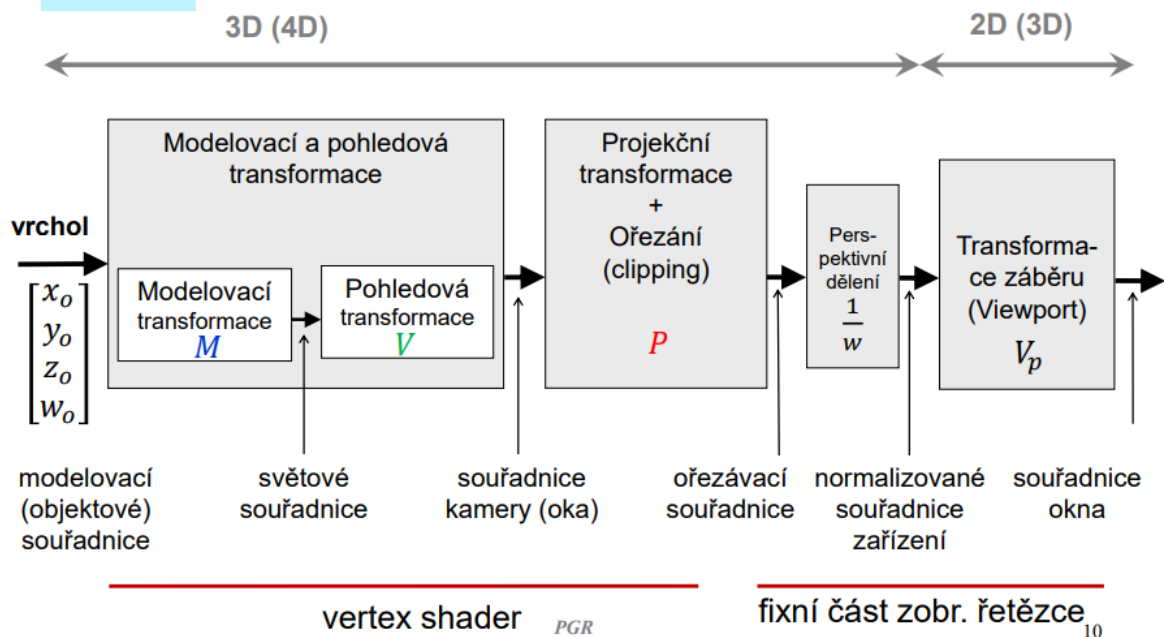
$$\begin{pmatrix} \frac{w}{2} & 0 & 0 & x + \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & y + \frac{h}{2} \\ 0 & 0 & \frac{f-n}{2} & \frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$o_x = x + \frac{w}{2}$   
 $o_y = y + \frac{h}{2}$   
 $z_w$  used for visibility test (Z-buffer)

PGR

53

**Posloupnost souřadných systémů, kterými prochází vrchol, než získá souřadnice v rámci okna.**



- objektové souřadnice - při modelování
- světové souřadnice - transformace ve světě
- souřadnice kamery/oka - vrcholy přesunuty do oka kamery
- ořezávací souřadnice - po projekci (ortho/persp) a clippingu
- NDC - Normalized Device Coordinates, po perspektivním dělení
- souřadnice okna - viewport transformace

## Transformace vůči zadanému souřadnému systému.

- definuje se maticí, světové souřadnice se vynásobí inverzní A maticí a pak A maticí, která definuje souřadný systém

## Matice rovnoběžného a perspektivního promítání.

- rovnoběžné (orthographic) - promítání do krychle, definováno 6 rovinami

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- perspektivní - ořízlá pyramida, definováno fovY, aspect ratio, near a far rovina

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

## Gimbal lock.

- problém v násobení všech os postupně, když prostřední rotace je 90° pak první a třetí rotace otáčejí podle stejné osy a jedna osa je tím pádem ztracena, řešení rotovat podle jedné custom osy -> kvaterniony

## Interpolace translace a rotace (kvaterniony, lerp a slerp).

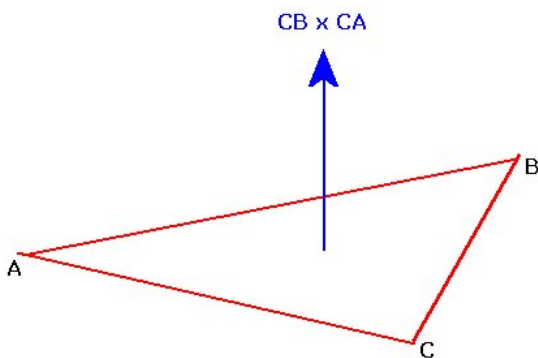
- $q = (\text{skalár}, i \text{ imag.}, j \text{ imag.}, k \text{ imag.})$ ,  $i$  - rotace x,  $j$  - rotace y,  $k$  - rotace z
- $i^2 = j^2 = k^2 = ijk = -1$
- sdružený kvaternion  $q = (q_0 + \mathbf{q})$ ,  $q^* = (q_0 - \mathbf{q})$
- norma  $|q| = q_0^2 + q_1^2 + q_2^2 + q_3^2$
- sčítání komutativní (per part)
- násobení nekomutativní  $pq = p_0q_0 - \mathbf{p} \cdot \mathbf{q} + p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q}$
- inverzní kvaternion  $q^{-1} = q^* / |q|^2$
- jednotkový kvaternion  $|q| = 1$
- rotace:
  - $q = \cos(\alpha) + \mathbf{uq} \sin(\alpha)$
  - $\alpha$  - úhel rotace
  - $\mathbf{uq}$  - osa rotace
- lerp -  $\text{lerp}(a, b, t) = (1 - t)a + bt$ , lineární interpolace, konstantní rychlost
- slerp - sférická interpolace, po kružnici, konstantní rychlost

## 4. Osvětlovací model

### Normálový vektor a jeho použití při výpočtu osvětlení v bodě.

- určují orientaci plochy, čímž udává kolik světla dostane z každého světla podle směru
- kolmý k ploše, per vertex
- určuje jak se paprsek světla odráží v daném bodě

### Výpočet normály trojúhelníka.



### Interpolace normály a ostatních vektorů a proč se používají normalizované vektory.

- různé velké trojúhelníky by měly různé velké normály a úhel mezi normály, který je potřeba by byl špatně vypočítán, v FS je potřeba znovu normalizovat protože během interpolace se zkrátí
- ?

### Phongův osvětlovací model, vzorce jednotlivých složek.

- lokální empirický model - žádné sekundární odrazy ani stíny
- 3 komponenty - ambientní, difúzní, spekulární
  - difúzní - směrové světlo, množství proporcionalní k úhlu mezi normálou a směrem k světlu

$$\text{diffuse}_{\text{reflected}} = \max(\cos \alpha, 0) * \text{diffuse}_{\text{light}} * \text{diffuse}_{\text{material}}$$

$$\cos \alpha = \vec{l} \cdot \vec{n}$$

- spekulární - odlesky, nejvíce vidět ve směru odlesku (reflection), shininess čím větší tím menší oblast odlesku

$$\text{specular}_{\text{reflected}} = [\max(\cos \beta, 0)]^{\text{shininess}_{\text{material}}} * \text{specular}_{\text{light}} * \text{specular}_{\text{material}}$$

$$\cos \beta = \vec{c} \cdot \vec{r}$$

$$\vec{r} = -\vec{l} + 2 \text{dot}(\vec{l}, \vec{n}) \vec{n}$$

- ambientní - stejná barva všude na modelu, velmi hrubá aproximace reality
  - $\text{ambi} = \text{ambi\_light} * \text{ambi\_material}$
- emisní - nemusí se používat, přidává odstín barvy
- finální barva je součtem vlivů všech světél, které jsou tlumená (např. vzdálenost), globální ambientní složky a emise

## Proč stačí kanály RGB (metamerismus)?

- Jev, kdy se dvě barvy lidskému zraku jeví jako stejné, přestože z hlediska spektrální charakteristiky stejné nejsou.
- 3 typy čípků - RGB

## Základní typy světél a jejich simulace ve Phongově osvětlovacím modelu.

- directional light - žádné tlumení, direction je ke světlu,  $\text{light} = (\text{ambi\_ref} + \text{diff\_ref} + \text{spec\_ref})$
- point light - plus má pozici a probíhá útlum podle vzdálenosti, linear, quadratic
  - $\text{LIGHT\_vec} = \text{norm}(\text{pos\_light} - \text{pos\_point})$
  - $\text{light} = \text{attenuation} * (...)$
- spot light - poloha, útlum a směr kuželu světla
  - osvětlení bodu podle  $\cos(\alpha) > \cos(\text{SPOT\_CUTOFF})$ , kde  $\alpha$  je úhel mezi pointToLight a spotlightDirection, SPOT\_CUTOFF je úhel mezi 0 a 90
  - $\text{light} = \text{spotlight\_effect} * \text{attenuation} * (...)$

## Metody stínování.

- flat shading - jedna barva na face, nerealistické
- Gouraud shading - světlo se počítá per vertex, interpolace barev podél vrcholů, odlesky (specular), jen ve vrcholech
- Phong shading - normálové vektory vrcholů jsou interpolovány do fragment shaderu, kde se počítá osvětlení

## 5. Základní parametrické křivky.

### Parametrická reprezentace.

- aktuální souřadnice podle parametru T (většinou čas), pohybuje se od 0 do 1
- snadno se dá určit bod na křivce
- lze použít ve 3D na rozdíl od explicitní a implicitní reprezentace
- parametrizace není jedinečná, různé parametrizace mohou popisovat stejnou křivku
- stupeň křivky + 1 = řád křivky
- stupeň 0 = jeden bod
- stupeň 1 = úsečka
- stupeň 2 = parabola
- stupeň 3 je ideální -> 4 stupně volnosti
- křivky jsou spojené v bodě (uzlu)

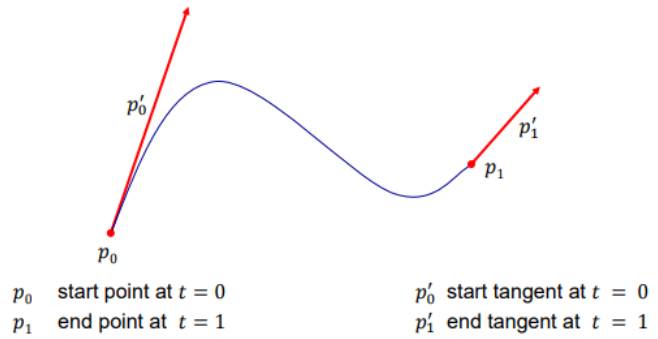
### Spojitosť při napojování segmentů.

- parametrická - směry a velikosti derivací jsou si rovny, C, implikuje geometrickou
- geometrická - směry derivací jsou si rovny ale velikosti nemusí, G, neimplikuje parametrickou
- C0 - stejný bod, křivky jsou spojené
- C1 - směr a rychlost jsou stejné, zrychlení se může prudce změnit, 1. derivace jsou si rovny
- C2 - směr, rychlost a zrychlení jsou stejné, 2. derivace jsou si rovny
- C2 implikuje C1 implikuje C0
- G0 - stejný jako C0
- G1 - křivka je hladká, směr se nemění, ale rychlost se může o násobek změnit, směry derivací jsou si rovny ale ne velikost
- G2 - směr a akcelerace jsou stejné ale jejich velikost se může změnit

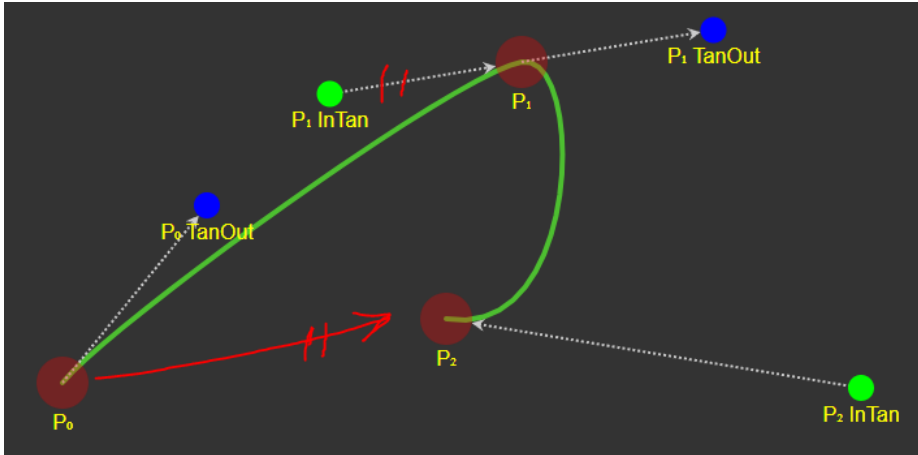
### Interpolační (Ferguson, Catmull-Rom) křivky.

- animační křivky, prochází kontrolními body, polynomiální (zbytečně komplikované, řešení rovnic) nebo parametrická, malá změna bodu -> velmi rozdílná křivka

- Ferguson (Hermite) - dány 2 body a jejich derivace, 4 báze funkce násobící body a derivace

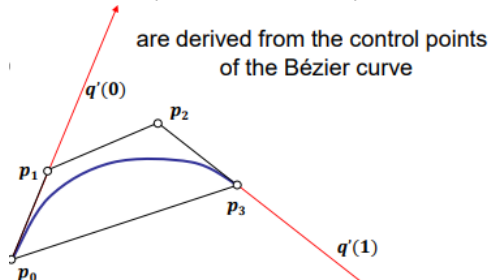


- více segmentů -> C1 pokud napojující body mají stejnou derivaci
- Catmull-Rom - kubická hermite křivka, časté v animacích pro kameru, C1, zadají se body a derivace jenom krajních bodů, derivace neokrajových bodů =  $(\text{next\_point} - \text{prev\_point}) / 2$



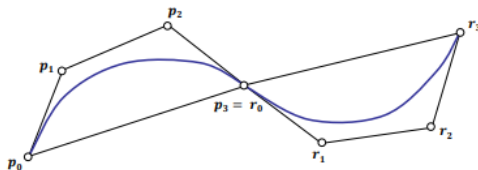
## Aproximační křivky (Bézier, B-spline, NURBS).

- modelovací křivky, používá vlivy kontrolních bodů, nemusí skrz ně procházet, součet báze funkcí ( $\geq 0$ ) je 1
- bezier - začíná v prvním bodě, končí v posledním, stupeň 3 cubic, bezier vs ferguson, fonty



- **bezier**

- pro výpočet buď báze funkce nebo de Casteljau
- C0 - poslední bod 1. křivky = první bod 2. křivky



C<sup>1</sup> continuity (parametric)

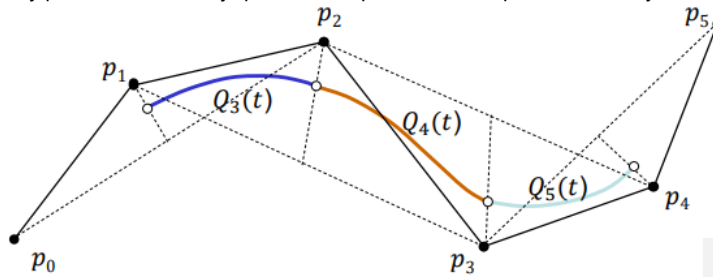
the curves touch at the join point       $p_3 = r_0$  (C<sup>0</sup> continuity)  
tangent vectors are the same       $p_3 - p_2 = r_1 - r_0$

- C1 -       $\Rightarrow$  point  $p_3 = r_0$  is in the middle of line  $p_2r_1$

- Coons - anti těžiště trojúhelníku 3 posloupných bodů viz obrázek b-spline, pokud uniformní se počítají báze funkce, pokud ne tak de-boor



- de-boor - výpočet báze funkcí (coons), pomocí interpolace báze funkcí
- B-spline - knot vector - udává váhu jednotlivých bodů, pro 4 body abcd je uniformní knot vector  $[0,1,2,3,4,5,6,7]$  - body + řád, úpravou lze křivku přiblížit nebo oddálit od bodu ale  $\rightarrow$  Non-Uniform
- Uniform Cubic B-Spline - vychází z Coons křivky, napojení několika coons segmentů, zajišťuje  $C^2$  spojitost, počítá se pomocí báze funkcí, pokud chceme aby procházelo bodem je potřeba ho opakovat nebo opakovat hodnoty knot vectoru  $[0,0,0,0,1,2,3,3,3,3]$ , pak už ale neplatí  $C^2$



arc $Q_3$	is defined by control points	$p_0, p_1, p_2, p_3$
	multiplied by basis functions	$C_0, C_1, C_2, C_3$
arc $Q_4$	is defined by control points	$p_1, p_2, p_3, p_4$
	multiplied by basis functions	$C_0, C_1, C_2, C_3$
arc $Q_5$	is defined by control points	$p_2, p_3, p_4, p_5$
	multiplied by basis functions	$C_0, C_1, C_2, C_3$

$n = 3$  degree  
 $k = 4$  order  
 $m = 6$  points  
 $u \in \langle 3, 6 \rangle$   
 $t \in \langle 0, 1 \rangle$  for each segment  
 class of continuity  $C^2$

- Non-Uniform Rational B-Spline curves - non-uniform znamená, že v knot vectoru nemusí být postupně čísla, body mají homogenní souřadnice  $[xw, yw, zw, w]$ , kde  $w$  je váha bodu, díky tomuto jsou invariantní k afinním transformacím a perspektivním projekcím, univerzální, možnost definovat přesně kuželosečky a válcové plochy, používá se hodně v modelování

## Adaptivní vykreslování Bézierovy křivky (algoritmus de Casteljau).

- rozdělí na 2 bezier křivky podle  $t$ , pokud je dostatečně rozdělená, spojíme nové kontrolní body

