Cvičení II

Pohyb kamery s využitím příkazu lookAt

Obsluha klávesnice

- zpětná volání (callbacky) pro obsluhu klávesnice
  - glutKeyboardFunc(keyboardCallback);
  - glutKeyboardUpFunc(keyboardUpCallback);
  - glutSpecialFunc(specialKeyboardCallback);
  - glutSpecialUpFunc(specialKeyboardUpCallback);

klávesy (ascii znaky)

funkční klávesy

- mapa kláves bool **keyMap**[KEYS_COUNT];
  - zaznamenáno, zda je klávesa stisknuta či uvolněna (true → stisknuta)
  - umožňuje testovat kombinace kláves
  - v callbacku časovače (timerCallback) volány obslužné funkce dle stisknutých kláves
  - frekvence opakování kláves závisí jen na frekvenci časovače

```
void specialKeyboardCallback(int specKeyPressed, int mouseX, int mouseY) {
   switch (specKeyPressed) {
      case GLUT_KEY_RIGHT:
         gameState.keyMap[KEY_RIGHT_ARROW] = true; break;
      …
   }
}
```

Transformace objektů

- každá instance objektu ve scéně má vlastní stavovou strukturu
  - odvozena ze struktury Object (např. *SpaceShipObject*)
  - instance všech objektů jsou uloženy v seznamech ve struktuře GameObjects definované na začátku asteroids.cpp (např. gameObjects.spaceShip)

render_stuff.h

```
typedef struct Object {
  glm::vec3 position;     // pozice objektu
  glm::vec3 direction;    // směr pohybu objektu
  float     speed;        // rychlost pohybu objektu
  float     size;         // velikost objektu
…
} Object;
typedef struct SpaceShipObject : public Object {
  float viewAngle;        // směr pohledu ve stupních
} SpaceShipObject;
```

- transformace objektu odvozena ze stavové struktury

```
glm::mat4 modelMatrix = glm::translate(glm::mat4(1.0f), spaceShip->position);
modelMatrix = glm::rotate(modelMatrix, glm::radians(spaceShip->viewAngle), glm::vec3(0, 0, 1));
modelMatrix = glm::scale(modelMatrix,
                         glm::vec3(spaceShip->size, spaceShip->size, spaceShip->size));
```

Transformace objektů

pohledová a projekční matice se předávají do kreslící funkce draw… a pomocí setTransformUniforms() se jako matice PVM předají do shaderů

```
void drawSpaceShip(SpaceShipObject *spaceShip, const glm::mat4& viewMatrix, const glm::mat4& projectionMatrix) {
    …
    glm::mat4 modelMatrix = …
    // send matrices to the vertex & fragment shader
    setTransformUniforms(modelMatrix, viewMatrix, projectionMatrix);
    …
}

void setTransformUniforms(const glm::mat4 &modelMatrix, const glm::mat4 &viewMatrix, const glm::mat4
&projectionMatrix) {
    glm::mat4 PVM = projectionMatrix * viewMatrix * modelMatrix;
    glUniformMatrix4fv(shaderProgram.PVMmatrixLocation, 1, GL_FALSE, glm::value_ptr(PVM));
}
```

vertex shader

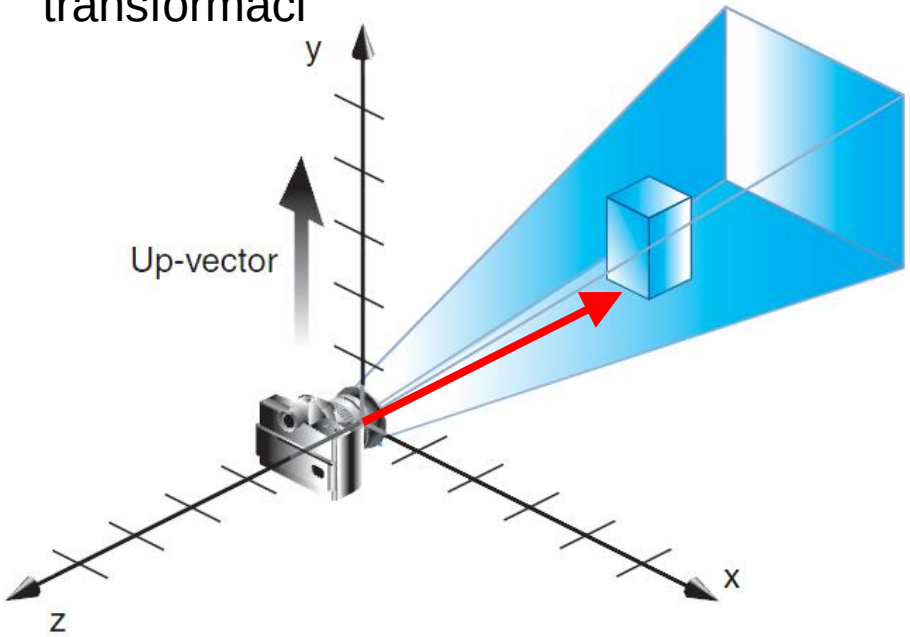
```
…
uniform mat4 PVMmatrix;
in vec3 position;

void main() {
    gl_Position = PVMmatrix * vec4(position, 1.0);
    …
}
```

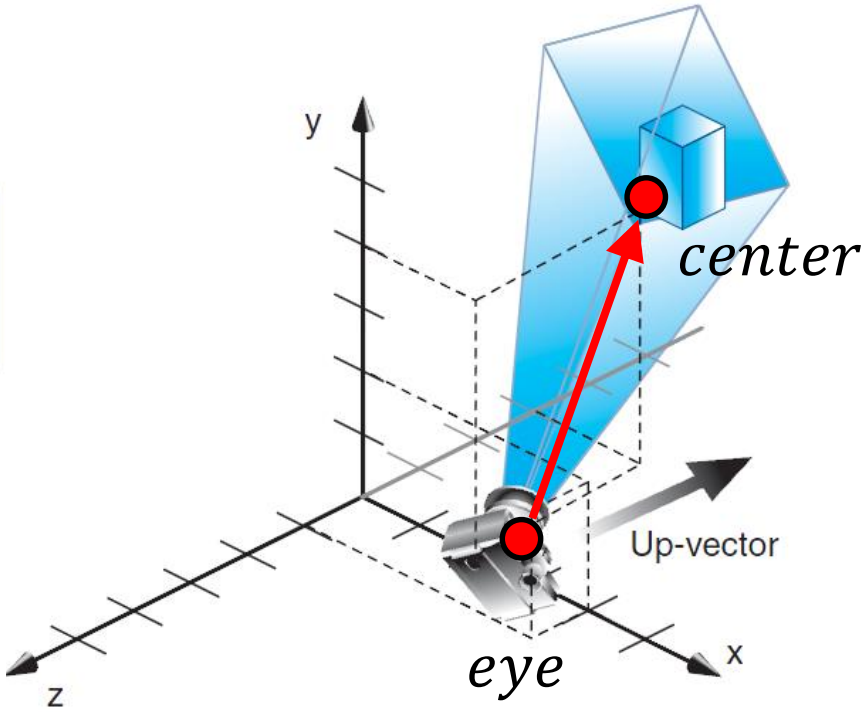
Pohledová transformace - lookAt

```
glm::mat4 viewMatrix = glm::lookAt(
    eye,                // pozice kamery
    center,             // bod na který kamera kouká
    upVector
  );
```

počáteční umístění kamery při jednotkové pohledové transformaci



jiné umístění kamery

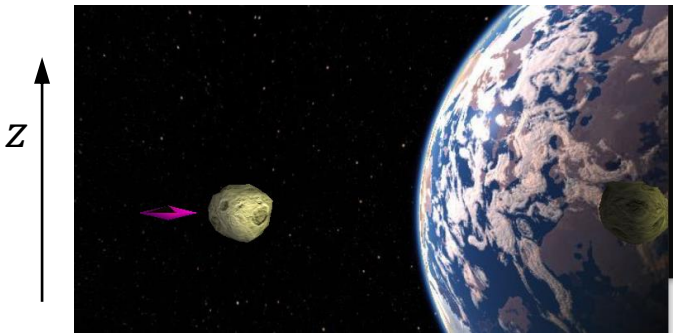


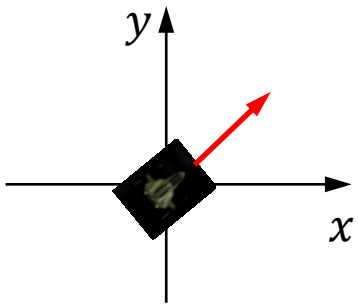
Úlohy

- implementace pohybu rakety v rovině xy
  - ovládání pomocí šipek

  task 2
- dynamický pohled z rakety
  - aktivace klávesou “c”
- rozhlížení kamerou nahoru a dolů pomocí myši *task 3*

dynamický pohled z rakety



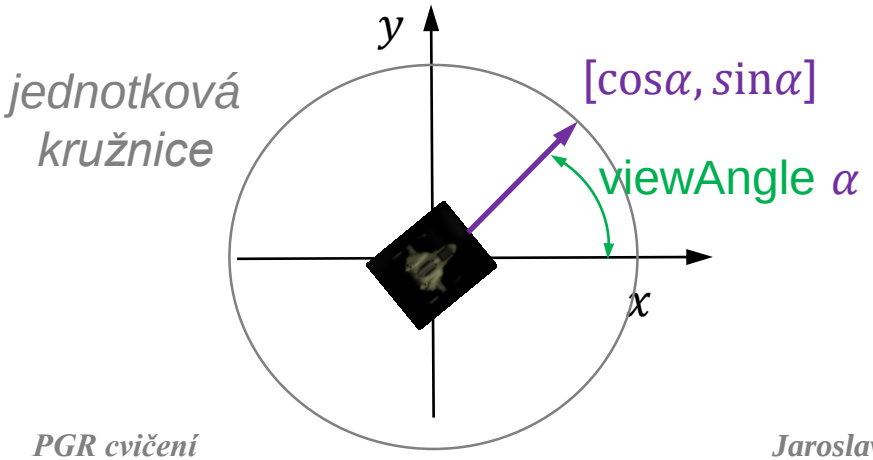


směr pohledu
=
směr natočení rakety v rovině xy (z=0)

Úloha 1 - pohyb rakety

DCGI

- ovládání pomocí šipek
  - ↑ zvýšení rychlosti rakety → funkce **increaseSpaceShipSpeed**()
  - ↓ snížení rychlosti rakety → funkce **decreaseSpaceShipSpeed**()
    - rychlost rakety → gameObjects.spaceShip->speed
  - ← otočení doleva → **turnSpaceShipLeft**()
  - → otočení doprava → **turnSpaceShipRight**()
    - změna úhlu natočení gameObjects.spaceShip->viewAngle [*stupně*]
      → úhel přepočítat na jednotkový vektor ve směru pohybu
      gameObjects.spaceShip->direction

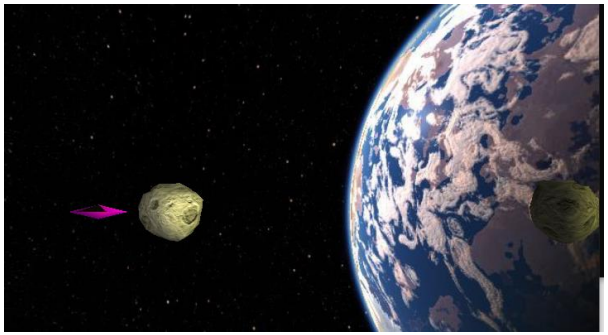


směr pohledu
=
směr natočení rakety v rovině xy (z=0)
→
spaceShip->direction

Úloha 1 - pohyb rakety

- aktualizace pozice rakety → funkce updateObjects()
  - posun ve směru pohledu gameObjects.spaceShip->direction
  - délka posunu rovna uražené dráze za daný čas

    s = v.t
- nastavení dynamického pohledu z rakety
  - pozice kamery → pozice rakety
  - směr pohledu $\overrightarrow{view}$ → směr pohybu rakety
  - raketa se pohybuje v rovině xy → $\overrightarrow{up}$ vektor dán osou z



aktivace dynamického pohledu z rakety

→

klávesa c

Úloha 2 - rozhlížení kamerou

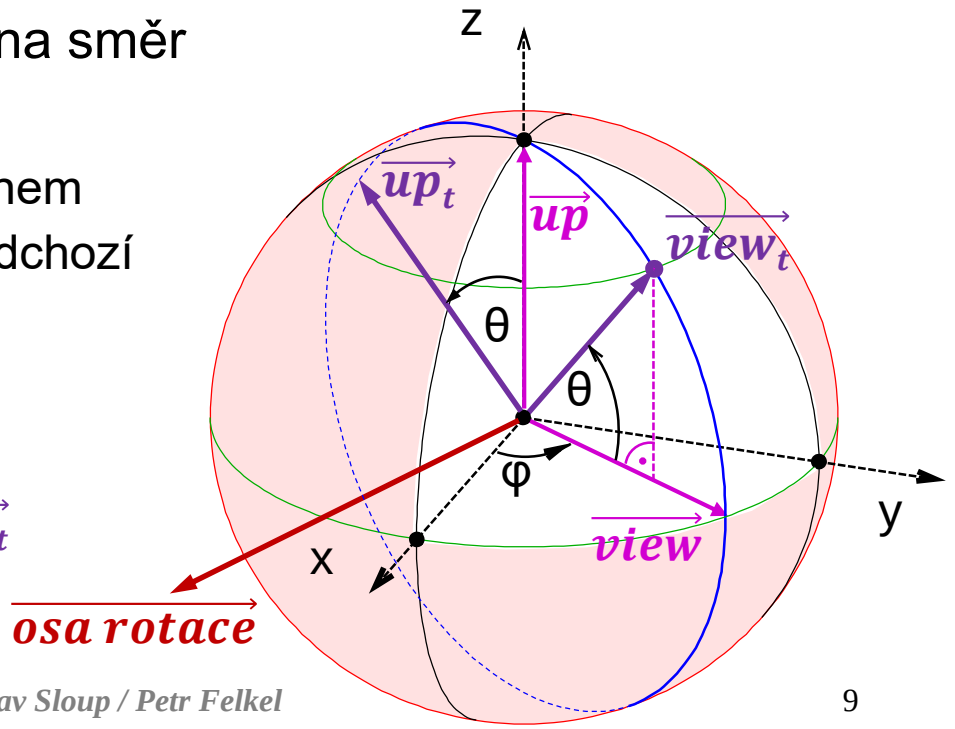
- rozhlížení kamerou nahoru a dolů pomocí myši

  → úhel gameState.cameraElevationAngle [*stupně*]

- možnosti implementace
  a) sférické souřadnice (*viz další slide*)
  b) rotace pohledu okolo osy kolmé na směr pohledu a up vektor

b) rotace pohledu okolo osy kolmé na směr pohledu a up vektor

- $\overrightarrow{osa\ rotace}$ dána vektorovým součinem vektorů $\overrightarrow{view}$ a $\overrightarrow{up}$ (tj. vektory z předchozí úlohy)
- otočením vektorů $\overrightarrow{view}$ a $\overrightarrow{up}$ okolo $\overrightarrow{osa\ rotace}$ získáme hledané transformované vektory $\overrightarrow{view_t}$ a $\overrightarrow{up_t}$
- rotační matice se získá voláním glm::rotate(…)



Úloha 2 a) – sférické souřadnice

- pozice P na jednotkové kouli ($r = 1$) umístěné v počátku

  → $P[x, y, z]$ = směr pohledu $\overrightarrow{view_t}$

  $$x = r \cos\varphi \cos\theta$$
  $$y = r \sin\varphi \cos\theta$$
  $$z = r \sin\theta$$

- up vektor

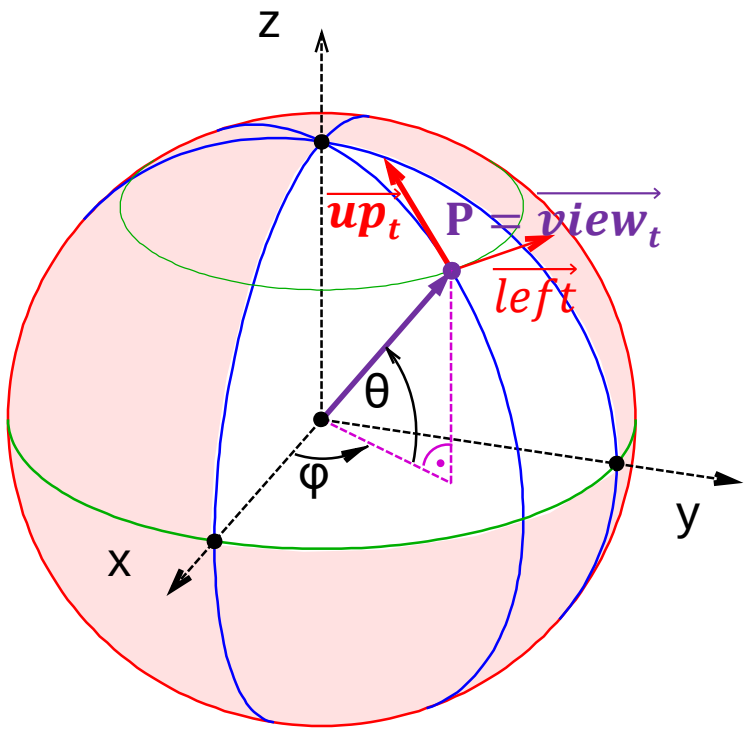
  → derivace P ve směru θ

  $$\overrightarrow{up_t} = \left[\frac{\partial x}{\partial\theta}, \frac{\partial y}{\partial\theta}, \frac{\partial z}{\partial\theta}\right]$$

- směr doleva

  → derivace P ve směru φ

  $$\overrightarrow{left} = \left[\frac{\partial x}{\partial\varphi}, \frac{\partial y}{\partial\varphi}, \frac{\partial z}{\partial\varphi}\right]$$



Užitečné rady

- převod na radiány glm::radians(uhelVeStupnich)
- parametr $center$ použitý při specifikaci pohledu pomocí lookAt je bod → směr pohledu se určí jako $\overrightarrow{view} = center - eye$
- soubor README.txt (*součást zipu projektu*) obsahuje popis, kam vkládat řešení jednotlivých úloh