FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
LABORATORIJ ZA INTERAKTIVNE SIMULACIJSKE SUSTAVE

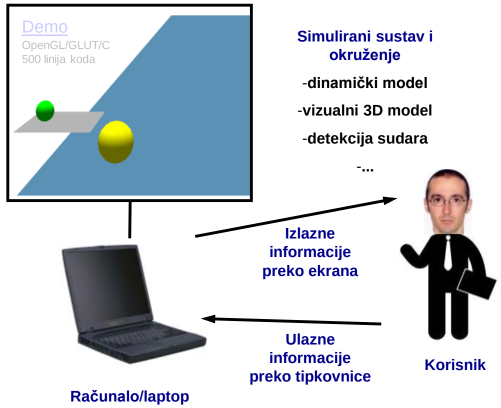
# Interaktivni simulacijski sustavi

## Matematičko modeliranje i numerička integracija kroz primjere pojednostavljenih interaktivnih simulacijskih sustava

Prof.dr.sc. Krešimir Ćosić
Dr.sc. Siniša Popović



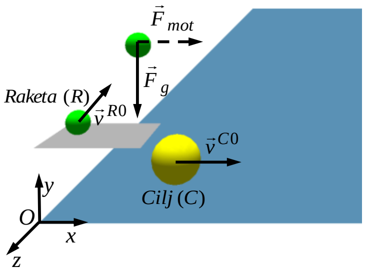
# Primjer interaktivnog simulacijskog sustava






# Dinamički model simuliranog sustava

Pretpostavke:
- raketa se giba u ravnini paralelnoj s *Oxy*
- cilj se giba paralelno s osi *Ox* po terenu koji se nalazi u ravnini $y = 0$
- postoji konstantna sila u smjeru osi *Ox* kada je uključen raketni motor




$$\vec{F}_g = m^R \vec{g},\ \vec{g} = const.$$

$$\vec{F}_{mot} = \begin{cases} m^R \vec{a}_{mot}, & \text{samo kada motor uklj.} \\ 0, & \text{inace} \end{cases},\quad \vec{a}_{mot} = const.$$

$$m^R \vec{a} = \vec{F}_g + \vec{F}_{mot} \Rightarrow \vec{a}(t) = \vec{g} + uklj(t)\vec{a}_{mot}$$

$$uklj(t) = \begin{cases} 1, & \text{ako je gumb 'm' za uklj./isklj. motora} \\ & \text{pritisnut neparni broj puta} \\ 0, & \text{inace} \end{cases}$$

**Model:**

Diferencijalne jednadžbe

Početni uvjeti diferencijalnih jednadžbi (stanje u početnom trenutku $t_0$)

$$\ddot{x}^R(t) = uklj(t) \cdot a_{mot},\ \dot{x}^R(t_0) = v^{xR0},\ x^R(t_0) = x^{R0}$$

$$\ddot{y}^R(t) = -g,\ \dot{y}^R(t_0) = v^{yR0},\ y^R(t_0) = y^{R0}$$

$$\dot{x}^C(t) = v^{C0},\ x^C(t_0) = x^{C0}$$

$$y^C(t) = y^{C0}$$

$$z^C(t) = z^R(t) = z^{RC0}$$

Konstantne funkcije za *y* i *z* koordinatu cilja te *z* koordinatu rakete, jer je tako zadano u pretpostavkama.
S obzirom da je tu sve riješeno, dalje ih ne spominjemo.



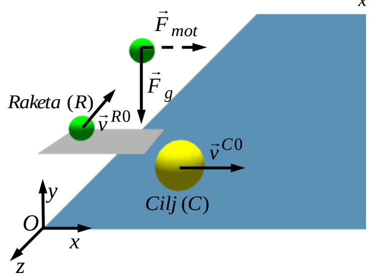
# Dinamički model simuliranog sustava

Model:

$$\ddot{x}^R(t) = uklj(t) \cdot a_{mot},\ \dot{x}^R(t_0) = v^{xR0},\ x^R(t_0) = x^{R0}$$

$$\ddot{y}^R(t) = -g,\ \dot{y}^R(t_0) = v^{yR0},\ y^R(t_0) = y^{R0}$$

$$\dot{x}^C(t) = v^{C0},\ x^C(t_0) = x^{C0}$$




**Kako računati položaj rakete i cilja u vremenu na način kao što se radi u simulatorima?**

Analitičko rješavanje? ... npr. na 3. jednadžbi:

$$\dot{x}^C(t) = v^{C0},\ x^C(t_0) = x^{C0} \Rightarrow x^C(t) = v^{C0}t + x^{C0} - v^{C0}t_0$$

Na računalu vrijeme teče u diskretnim koracima $t_0$, $t_0+T$, $t_0+2T$, $t_0+3T$,... pa *x* koordinatu cilja u tim trenucima dobijemo iz analitičkog rješenja:

$$x_n^C := x^C(t_n) = v^{C0} \cdot t_n + x^{C0} - v^{C0}t_0$$

$$t_n = t_{n-1} + T = t_0 + nT$$

**Međutim analitičko rješavanje diferencijalnih jednadžbi ne skalira na sve praktično važne simulacijske probleme.**

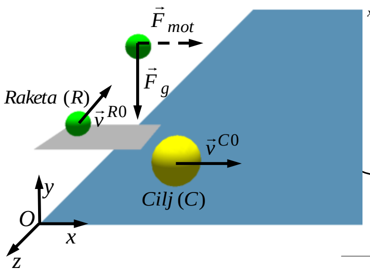
**Stoga se upotrebljava numerička integracija.**

# Dinamički model simuliranog sustava

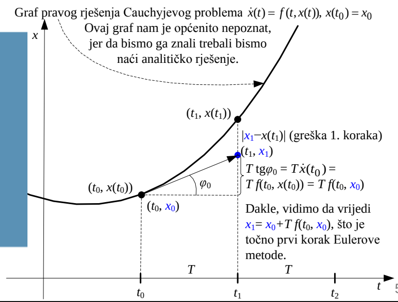
**Jednostavan primjer numeričke integracije – Eulerova metoda za rješavanje sljedećih diferencijalnih jednadžbi prvog reda s početnim uvjetima:**

$\dot{x}(t) = f(t, x(t)),\ x(t_0) = x_0$ tzv. *Cauchyjev problem* (engl. *initial value problem*) – npr. $\dot{x}^C(t) = v^{C0},\ x(t_0) = x^{C0}$

***Metoda:*** $x_n = x_{n-1} + T f(t_{n-1}, x_{n-1})$, za $n = 1, 2, 3, \ldots$, uz $x_0 = x(t_0)$ poznat iz početnog uvjeta, pri čemu je:
$t_n = t_{n-1} + T$, za $n = 1, 2, \ldots$, a $t_0$ je također poznat iz početnog uvjeta.
Varijabla $T$ predstavlja korak Eulerove metode.




**Eulerova metoda temelji se na lokalnoj aproksimaciji tangentom**





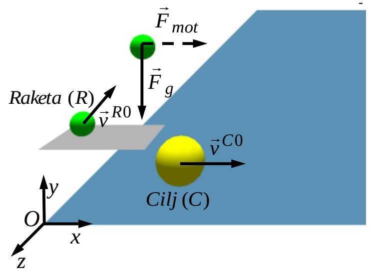

# Dinamički model simuliranog sustava

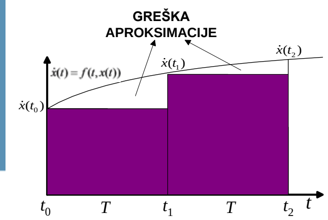
**Jednostavan primjer numeričke integracije – Eulerova metoda za rješavanje sljedećih diferencijalnih jednadžbi prvog reda s početnim uvjetima:**

$\dot{x}(t) = f(t, x(t)),\ x(t_0) = x_0$ tzv. *Cauchyjev problem* (engl. *initial value problem*) – npr. $\dot{x}^C(t) = v^{C0},\ x(t_0) = x^{C0}$

***Metoda:*** $x_n = x_{n-1} + T f(t_{n-1}, x_{n-1})$, za $n = 1, 2, 3, \ldots$, uz $x_0 = x(t_0)$ poznat iz početnog uvjeta, pri čemu je:
$t_n = t_{n-1} + T$, za $n = 1, 2, \ldots$, a $t_0$ je također poznat iz početnog uvjeta.
Varijabla $T$ predstavlja korak Eulerove metode.

**Drugačiji način gledanja na Eulerovu metodu:**
- Tražena funkcija $x(t)$ je dakako integral funkcije $\dot{x}(t)$
- Eulerovom metodom integral funkcije $\dot{x}(t)$ iterativno se računa tako da se aproksimira sa sumom površina pravokutnika ispod grafa ove funkcije, tj. $x_n = x_{n-1} + T\dot{x}(t_{n-1})$






# Dinamički model simuliranog sustava

**Jednostavan primjer numeričke integracije – Eulerova metoda za rješavanje sljedećih diferencijalnih jednadžbi prvog reda s početnim uvjetima:**

$\dot{x}(t) = f(t, x(t)),\ x(t_0) = x_0$ tzv. *Cauchyjev problem* (engl. *initial value problem*) – npr. $\dot{x}^C(t) = v^{C0},\ x(t_0) = x^{C0}$

***Metoda:*** $x_n = x_{n-1} + T f(t_{n-1}, x_{n-1})$, za $n = 1, 2, 3, \ldots$, uz $x_0 = x(t_0)$ poznat iz početnog uvjeta, pri čemu je:
$t_n = t_{n-1} + T$, za $n = 1, 2, \ldots$, a $t_0$ je također poznat iz početnog uvjeta.
Varijabla $T$ predstavlja korak Eulerove metode.

**Digresija na jedan primjer:** Inkrementalno računanje funkcije $x(t)$ kao u interaktivnoj simulaciji, uz primjenu Eulerove metode s korakom $T = 0.01$ s, ako je funkcija $x(t)$ definirana na sljedeći način:

$$\dot{x}(t) = t + x^2(t),\ x(1) = 2$$

$$x_0 = 2,\ t_0 = 1,$$

$$x_1 = x_0 + T(t_0 + x_0^2) = 2 + 0.01(1 + 2^2) = 2.05,\ t_1 = t_0 + T = 1.01$$

$$x_2 = x_1 + T(t_1 + x_1^2) = 2.05 + 0.01(1.01 + 2.05^2) = 2.102125,\ t_2 = t_1 + T = 1.02$$

...



# Dinamički model simuliranog sustava

Model:

1. $\ddot{x}^R(t) = uklj(t) \cdot a_{mot},\ \dot{x}^R(t_0) = v^{xR0},\ x^R(t_0) = x^{R0}$
2. $\ddot{y}^R(t) = -g,\ \dot{y}^R(t_0) = v^{yR0},\ y^R(t_0) = y^{R0}$
3. $\dot{x}^C(t) = v^{C0},\ x^C(t_0) = x^{C0}$

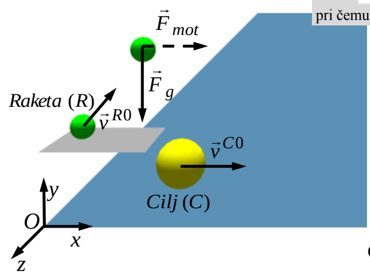
**Kako izgleda Eulerova metoda za diferencijalnu jednadžbu višeg reda, kao što su 1. ili 2. diferencijalna jednadžba modela?**

**1.** Svođenje na sustav diferencijalnih jednadžbi prvog reda uvođenjem novih varijabli. Sustav dobiva oblik:

$$\dot{x}(t) = f(t, x(t)),\ x(t_0) = x_0$$

pri čemu su $x(t)$ i $f(t, x(t))$ vektori.

$$\dot{v}^{xR} = uklj(t) \cdot a_{mot}$$
$$\dot{x}^R = v^{xR}$$
$$\dot{v}^{yR} = -g$$
$$\dot{y}^R = v^{yR}$$
$$\dot{x}^C = v^{C0}$$




**2.** Primjena Eulerove metode na svaku jednadžbu prvog reda u dobivenom sustavu: $x_n = x_{n-1} + T f(t_{n-1}, x_{n-1})$

$$v_n^{xR} = v_{n-1}^{xR} + T \cdot uklj(t_{n-1}) \cdot a_{mot}$$
$$x_n^R = x_{n-1}^R + T \cdot v_{n-1}^{xR}$$
$$v_n^{yR} = v_{n-1}^{yR} - T \cdot g$$
$$y_n^R = y_{n-1}^R + T \cdot v_{n-1}^{yR}$$
$$x_n^C = x_{n-1}^C + T \cdot v^{C0}$$

**Ove jednadžbe su programski ostvarene u demou na početku predavanja.**

# Dinamički model simuliranog sustava

```
state.gravity = 1;
state.t_step = 0.025; // seconds
void executeMissileMotionStep()
{
state.missile.vx_n = state.missile.vx_nminus1 +
    state.t_step * state.missile.isThrustOn * state.missile.thrust;
state.missile.x_n = state.missile.x_nminus1 +
    state.t_step * state.missile.vx_nminus1;
state.missile.vy_n = state.missile.vy_nnminus1 -
    state.t_step * state.gravity;
state.missile.y_n = state.missile.y_nminus1 +
    state.t_step * state.missile.vy_nminus1

state.missile.x_nminus1 = state.missile.x_n;
state.missile.y_nminus1 = state.missile.y_n;
state.missile.vx_nminus1 = state.missile.vx_n;
state.missile.vy_nminus1 = state.missile.vy_n;
}

void executeTargetMotionStep()
{
state.tgt.x_n = state.tgt.x_nminus1 + state.tgt.v0 * state.t_step;
state.tgt.x_nminus1 = state.tgt.x_n;
}
```

$$v_n^{xR} = v_{n-1}^{xR} + T \cdot uklj(t_{n-1}) \cdot a_{mot}$$
$$x_n^R = x_{n-1}^R + T \cdot v_{n-1}^{xR}$$

$$v_n^{yR} = v_{n-1}^{yR} - T \cdot g$$
$$y_n^R = y_{n-1}^R + T \cdot v_{n-1}^{yR}$$

$$x_n^C = x_{n-1}^C + T \cdot v^{C0}$$

**Ove jednadžbe su programski ostvarene u demou na početku predavanja.**



# Dinamički model simuliranog sustava

**Digresija na primjer rješavanja sustava diferencijalnih jednadžbi prvog reda Eulerovom metodom:**
Inkrementalno računanje funkcija $x(t)$ i $y(t)$ kao u interaktivnoj simulaciji, uz primjenu Eulerove metode s korakom $T = 0.01$ s, ako su funkcije $x(t)$ i $y(t)$ definirane na sljedeći način:

$$\dot{x}(t) = 2t + x(t)y(t),\ x(0) = 1$$
$$\dot{y}(t) = ty(t) + x(t),\ y(0) = -1$$

$x_0 = 1,\ y_0 = -1, t_0 = 0,$

$$\begin{cases} x_1 = x_0 + T(2t_0 + x_0 y_0) = 1 + 0.01(2 \cdot 0 + 1 \cdot (-1)) = 0.99, \\ y_1 = y_0 + T(t_0 y_0 + x_0) = -1 + 0.01(0 \cdot (-1) + 1) = -0.99, \\ t_1 = t_0 + T = 0.01 \end{cases}$$

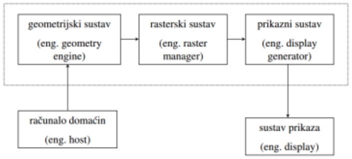
$$\begin{cases} x_2 = x_1 + T(2t_1 + x_1 y_1) = 0.99 + 0.01(2 \cdot 0.01 + 0.99 \cdot (-0.99)) = 0.980399, \\ y_2 = y_1 + T(t_1 y_1 + x_1) = -0.99 + 0.01(0.01 \cdot (-0.99) + 0.99) = -0.980199, \\ t_2 = t_1 + T = 0.02 \end{cases}$$

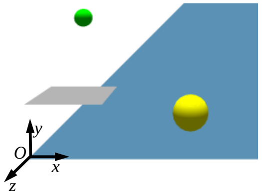
...



# Vizualni 3D model simuliranog sustava i okruženja

- Sadrži podatke koji se šalju u grafički protočni sustav



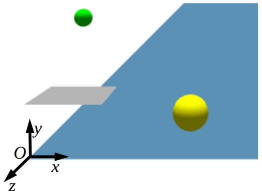




  - Definiranje geometrije pojedinih objekata
  - Definiranje materijala od kojih su sačinjeni objekti
  - Dodavanje svjetla u scenu
  - Pozicioniranje promatrača/kamere u sceni
  - Organiziranje scene – međusobni razmještaj objekata
  - ...

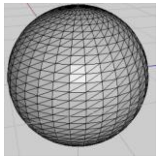


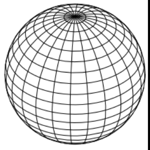
# Vizualni 3D model simuliranog sustava i okruženja

- geometrija se tipično predstavlja mrežama poligona/trokuta jer je grafički protočni sustav sklopovski prilagođen ovoj reprezentaciji
- trokut je posebno prikladan
  - svi vrhovi mu uvijek leže u istoj ravnini
  - uvijek je konveksan poligon

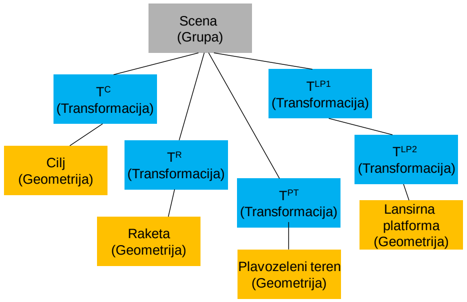


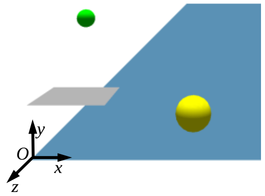

n // broj vrhova
m // broj trokuta
$x_1$ $y_1$ $z_1$ // decimalne vrijednosti koordinata vrhova
...
$x_n$ $y_n$ $z_n$
$v_{1,1}$ $v_{1,2}$ $v_{1,3}$ // cjelobrojni indeksi vrhova
...
$v_{m,1}$ $v_{m,2}$ $v_{m,3}$





# Vizualni 3D model simuliranog sustava i okruženja




$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

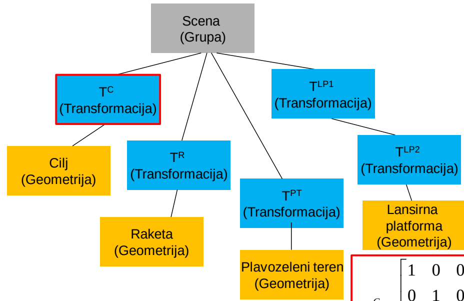
$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

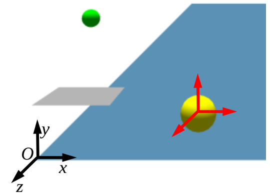
Razmještaj se logički može promatrati sukladno konceptu **grafa scene**
- hijerarhijska stablasta organizacija
- razne vrste čvorova – grupirajući, transformacijski, geometrijski
- geometrijski čvorovi su u listovima stabla

**Napomena**: Graf scene u smislu odgovarajuće strukture podataka nije bio programski ostvaren.



# Vizualni 3D model simuliranog sustava i okruženja




**Matrica transformacije cilja tijekom iscrtavanja scene za trenutak $t_n = t_0 + nT$ (translacija):**

$$x_n^C = x_{n-1}^C + T \cdot v^{C0}$$
$$y_n^C = y^{C0}$$
$$z_n^C = z^{RC0}$$

$$\mathbf{T}_n^C = \begin{bmatrix} 1 & 0 & 0 & x_n^C \\ 0 & 1 & 0 & y_n^C \\ 0 & 0 & 1 & z_n^C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C_n^{Scena} = \mathbf{T}_n^C C$$

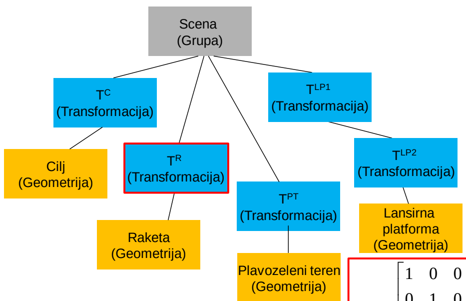
$C$ je geometrija cilja s obzirom na lokalni koordinatni sustav (tj. centar žute sfere)

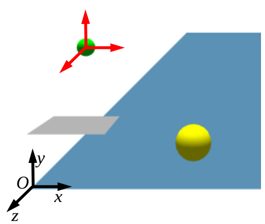
$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Vizualni 3D model simuliranog sustava i okruženja




**Matrica transformacije rakete tijekom iscrtavanja scene za trenutak $t_n = t_0 + nT$ (translacija):**

$$v_n^{xR} = v_{n-1}^{xR} + T \cdot uklj(t_{n-1}) \cdot a_{mot}$$
$$x_n^R = x_{n-1}^R + T \cdot v_{n-1}^{xR}$$
$$v_n^{yR} = v_{n-1}^{yR} - T \cdot g$$
$$y_n^R = y_{n-1}^R + T \cdot v_{n-1}^{yR}$$
$$z_n^R = z^{RC0}$$

$$\mathbf{T}_n^R = \begin{bmatrix} 1 & 0 & 0 & x_n^R \\ 0 & 1 & 0 & y_n^R \\ 0 & 0 & 1 & z_n^R \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_n^{Scena} = \mathbf{T}_n^R R$$

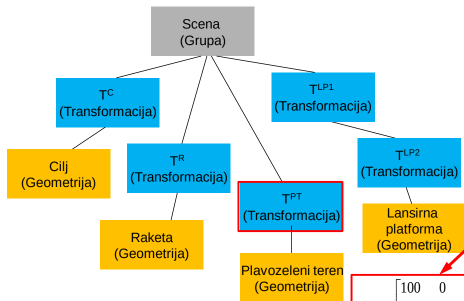
$R$ je geometrija rakete s obzirom na lokalni koordinatni sustav (tj. centar zelene sfere)

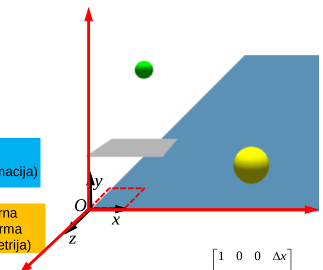
$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



# Vizualni 3D model simuliranog sustava i okruženja




**Matrica transformacije plavozelenog terena tijekom iscrtavanja scene za trenutak $t_n = t_0 + nT$ (konstantno skaliranje):**

$$\mathbf{T}_n^{PT} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

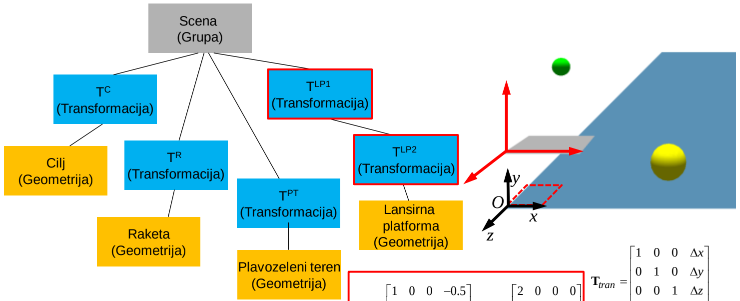
$$PT_n^{Scena} = \mathbf{T}_n^{PT} PT$$

$PT$ je geometrija crveno iscrtkane krpice koja seže do 1 po $x$-osi i do −1 po $z$-osi

$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## Vizualni 3D model simuliranog sustava i okruženja




**Matrica transformacije lansirne platforme tijekom iscrtavanja scene za trenutak $t_n = t_0 + nT$ (konstantna translacija i konstantno skaliranje):**

$$\mathbf{T}_n^{LP1} = \begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_n^{LP2} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$LP_n^{SC} = \mathbf{T}_n^{LP1}\mathbf{T}_n^{LP2} LP$$

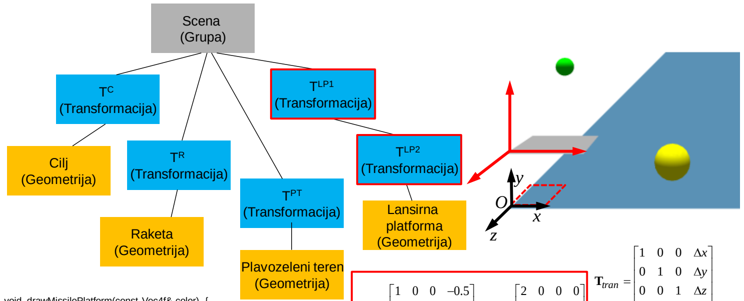
$LP$ je geometrija crveno iscrtkane krpice koja je identična kao prethodna za teren.

$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

17

## Vizualni 3D model simuliranog sustava i okruženja




```
void  drawMissilePlatform(const  Vec4f& color)  {
  glPushMatrix();
  glTranslated(-0.5, state.missilePlatformHeight, -1.0);
  glScaled(2.0, 2.0, 2.0);
  glMaterialfv(GL_FRONT, GL_AMBIENT, color.coords);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, color.coords);

  glBegin(GL_TRIANGLES); //Begin coordinates
    ...
  glEnd(); //End coordinates
  glPopMatrix();   }
```

$$\mathbf{T}_n^{LP1} = \begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_n^{LP2} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$LP_n^{SC} = \mathbf{T}_n^{LP1}\mathbf{T}_n^{LP2} LP$$

$LP$ je geometrija crveno iscrtkane krpice koja je identična kao prethodna za teren.

$$\mathbf{T}_{tran} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{skal} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

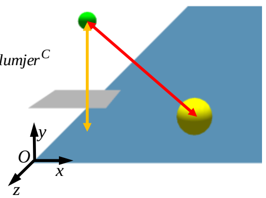
18

## Detekcija sudara

- sudar raketa-teren: $y_n^R \le polumjer^R$
- sudar raketa-cilj:

$$\sqrt{(x_n^R - x_n^C)^2 + (y_n^R - y_n^C)^2 + (z_n^R - z_n^C)^2} \le polumjer^R + polumjer^C$$

- ne provjerava se sudar rakete i lansirne platforme

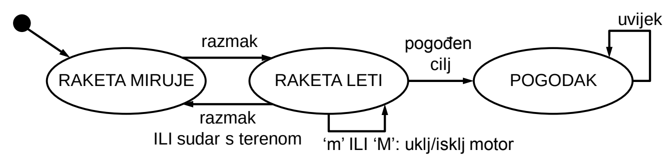



- ove jednostavne metode za detekciju sudara prilagođene su ovoj vrsti scene, jer npr. implicitno pretpostavljaju:
  - da su raketa i cilj dovoljno spori
  - da raketu lansiramo u smjeru pozitivne osi *Ox*, a ne recimo u suprotnom smjeru gdje teren ne postoji

19

## Obrada ulaznih informacija

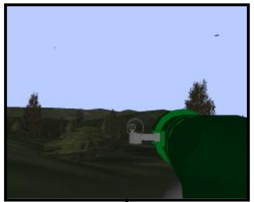
- ulazi se pojavljuju aperiodički - bitna razlika naspram:
  - približno periodičkog izvođenja numeričke integracije dinamičkog modela rakete i cilja s korakom $T$
  - isrtavanja koje mora reflektirati te promjene u prikazu scene
- dodatno, aperiodički se pojavljuje i sudar
- potreban je stroj stanja za reagiranje na aperiodičke informacije (ulazi i info o sudaru)




20

# Sve opisano postoji i u simulatorima, ali je bitno veće složenosti

**Npr. u simulatoru raketnog sustava za protuzračnu obranu**



**Simulirani sustav i okruženje**

**-dinamički model**

**-vizualni 3D model**

**-detekcija sudara**

**-...**

**Računalna infrastruktura**

**Informacije preko izlaznih uređaja**

**Informacije preko ulaznih uređaja**

**Operator sustava**