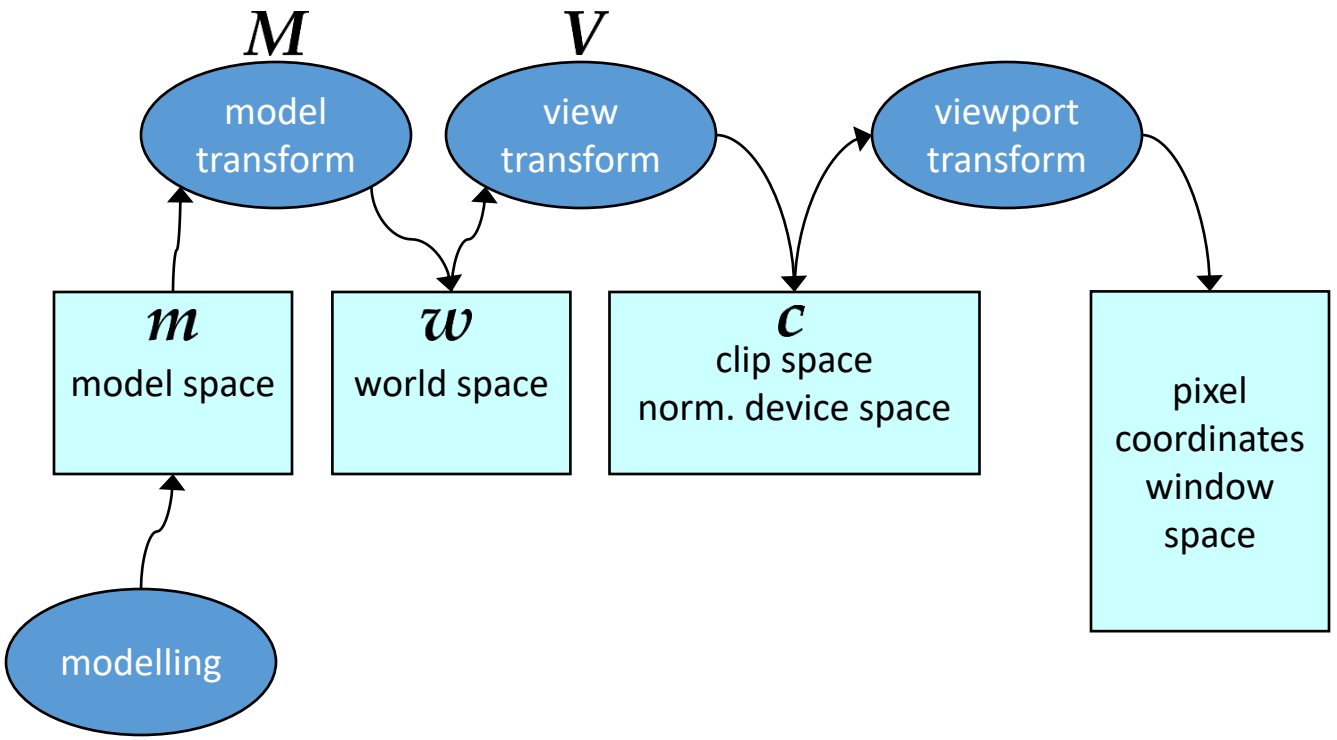
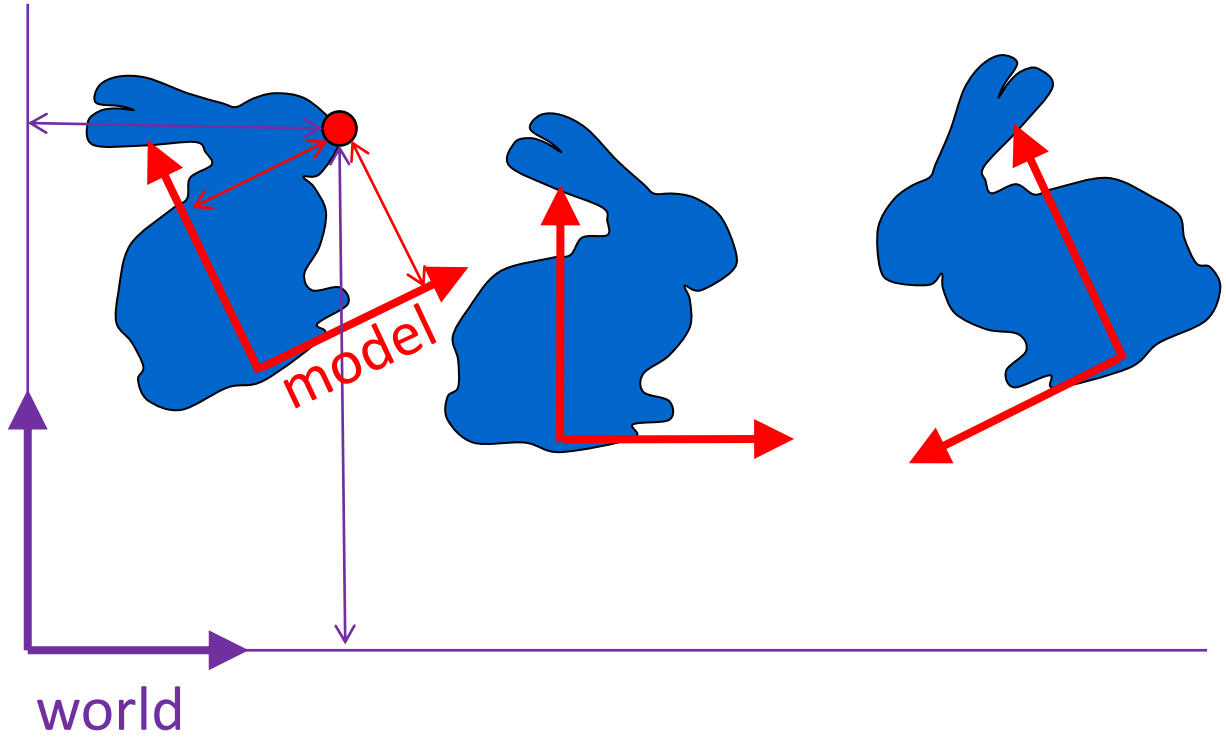
2D camera

László Szécsi

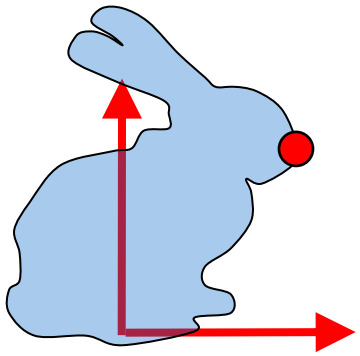
Transformation pipeline



Model and world coordinates (static interpretation)



Model and world coordinates (dynamic interpretation)



Constructing a model transformation

$$\begin{bmatrix} w_x & w_y & 1 \end{bmatrix} =$$

$$\begin{bmatrix} m_x & m_y & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ q_x & q_y & 1 \end{bmatrix}$$

from vertex buffer

Constructing a model transformation

$$\begin{pmatrix} w_x & w_y & 1 \end{pmatrix} =$$

$$\begin{pmatrix} m_x & m_y & 1 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ q_x & q_y & 1 \end{bmatrix}$$

from vertex buffer

new Mat4().scale(sx, sy).rotate(phi).translate(qx, qy)

compose matrix on host

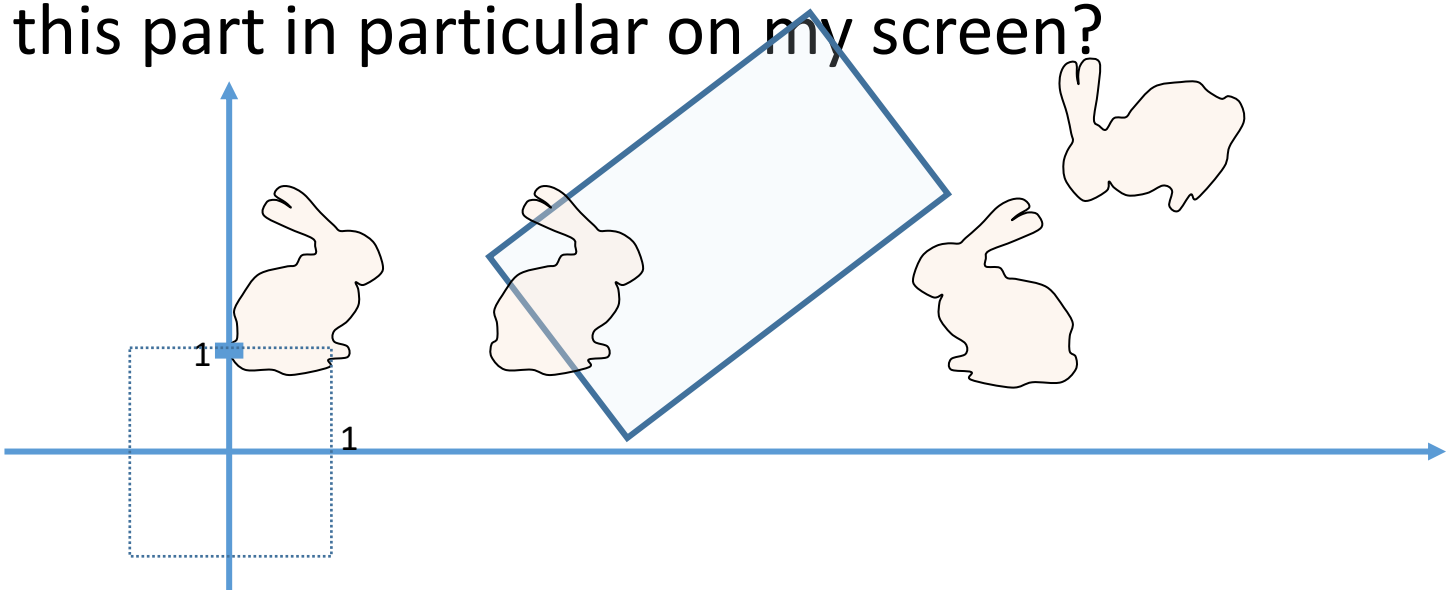
vector-matrix multiplication happens in the shader

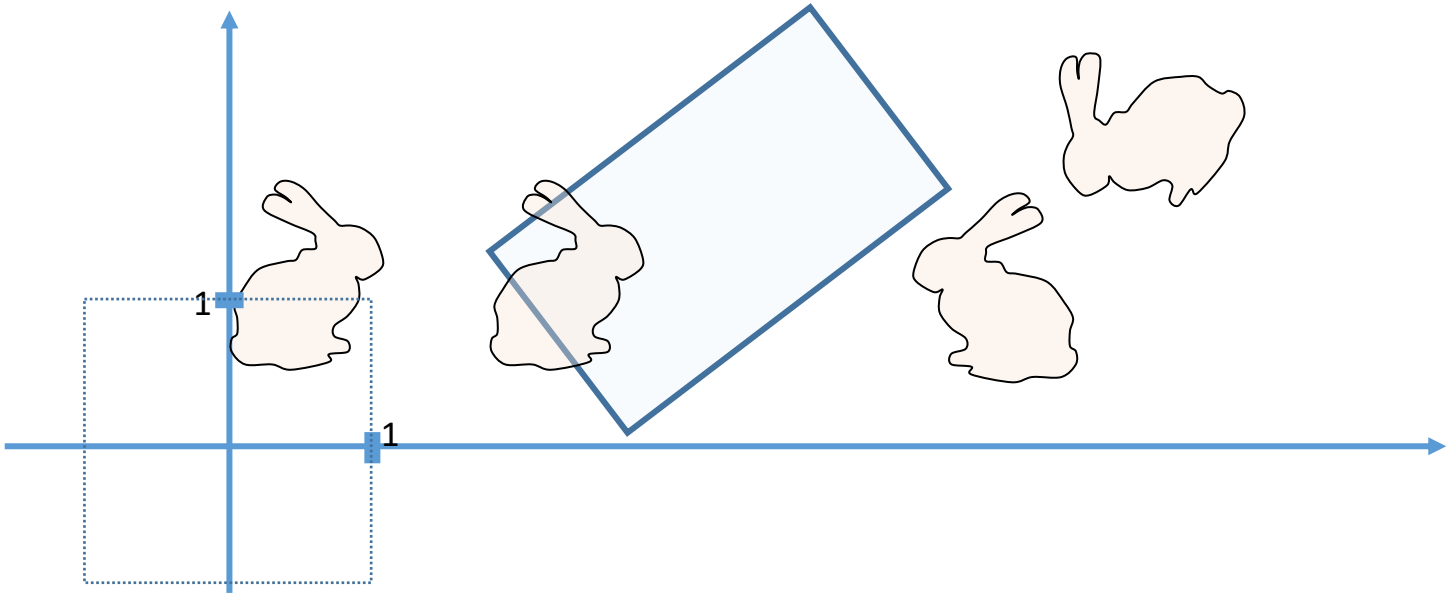
View transformation – 2D camera, OrthoCamera

- Where does this vertex go on the screen?

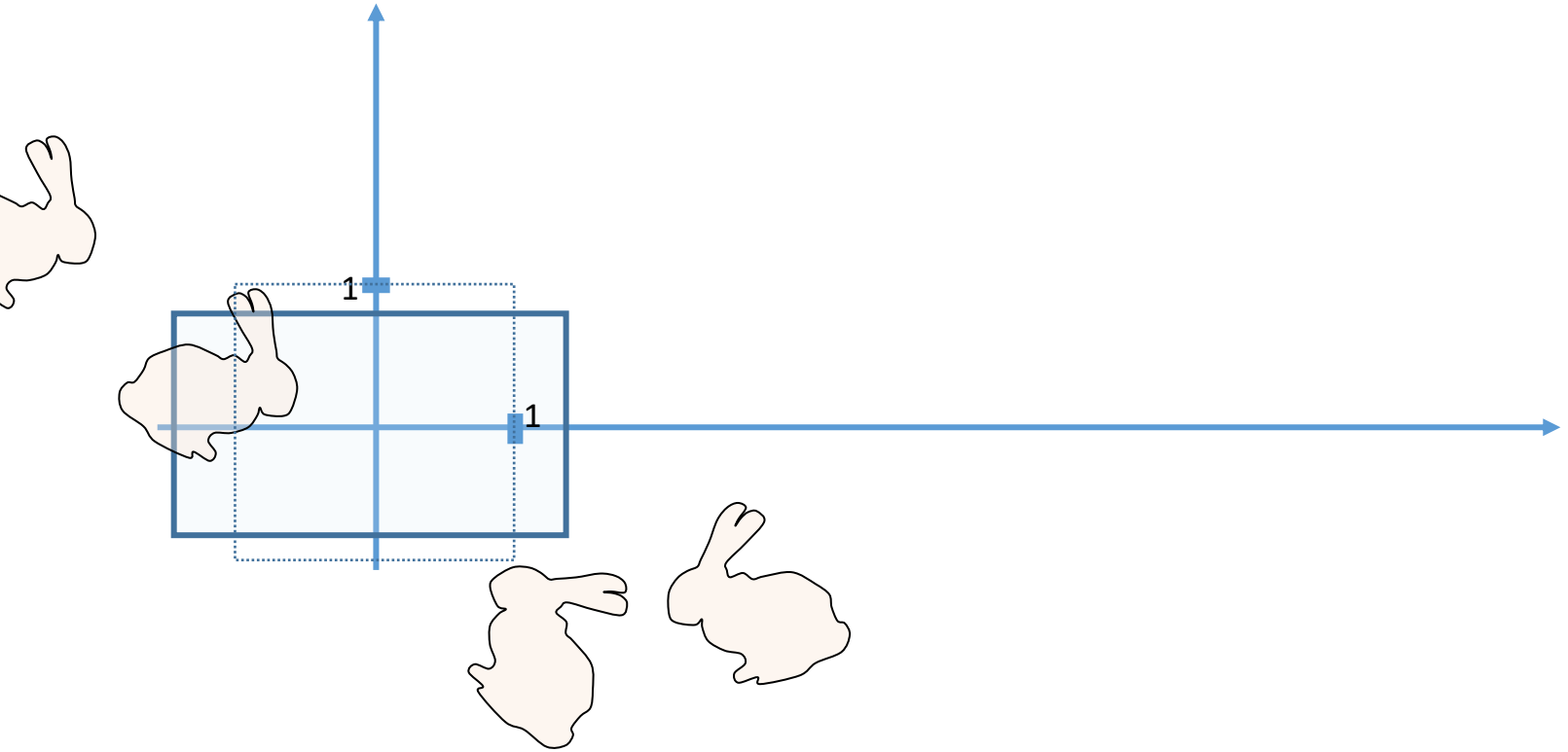
  also:

- What do I do with everything in the world, to get this part in particular on my screen?

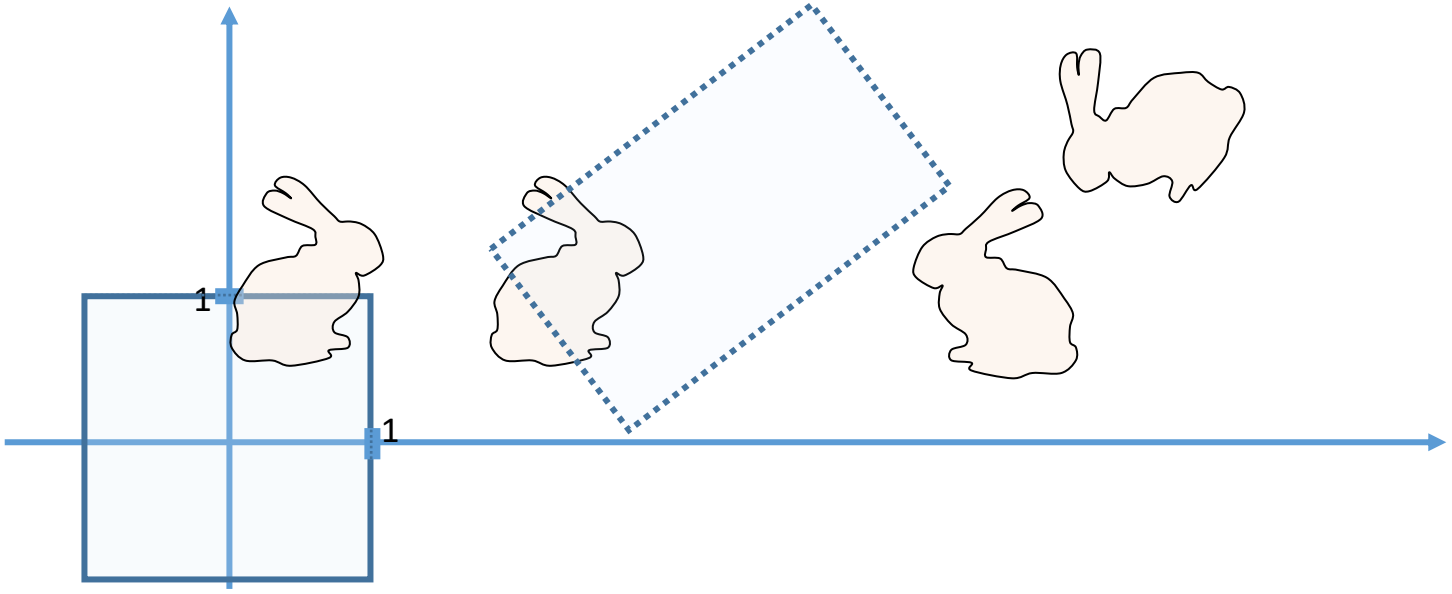




$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}}$$



$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}} \boldsymbol{S}_{\text{view}}$$



$$\begin{pmatrix} c_x & c_y & 1 \end{pmatrix} = \begin{pmatrix} w_x & w_y & 1 \end{pmatrix} \boldsymbol{T}_{\text{view}} \boldsymbol{R}_{\text{view}} \boldsymbol{S}_{\text{view}} \qquad \left(\frac{1}{2} \boldsymbol{S}_{\text{windowsize}} \boldsymbol{R}_{\text{windowangle}} \boldsymbol{T}_{\text{windowpos}} \right)^{-1}$$

OrthoCamera

```
class OrthoCamera() : UniformProvider("camera") {
  val position = Vec2(0.0f, 0.0f)
  val roll = 0.0f
  val windowSize = Vec2(2.0f, 2.0f)

  val viewProjMatrix by Mat4()
  init{
    updateViewProjMatrix()
  }
}
```

OrthoCamera::
updateViewProjMatrix

```
fun updateViewProjMatrix() {
  viewProjMatrix.set().
    scale(0.5f, 0.5f, 1.0f).
    scale(windowSize).
    rotate(roll).
    translate(position).
    invert()
}
fun setAspectRatio(ar : Float) {
  windowSize.x = windowSize.y * ar
  updateViewProjMatrix()
}
```

Rajzolás a kamerával

```
val camera = OrthoCamera()
```

```
gameObjects.forEach{ it.draw( camera ) }
```

Scene mint UniformProvider

```
class Scene (
  val gl : WebGL2RenderingContext) :
                    UniformProvider("scene"){



addComponentsAndGatherUniforms()

gameObjects.forEach{ it.draw( this, camera )
}
```