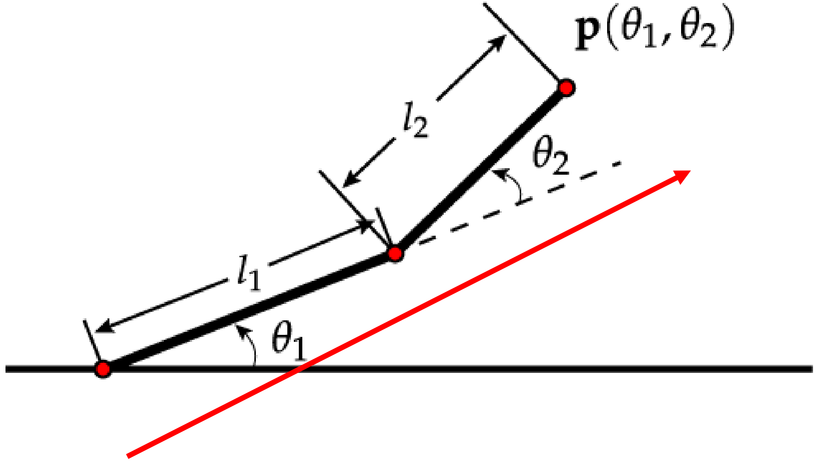
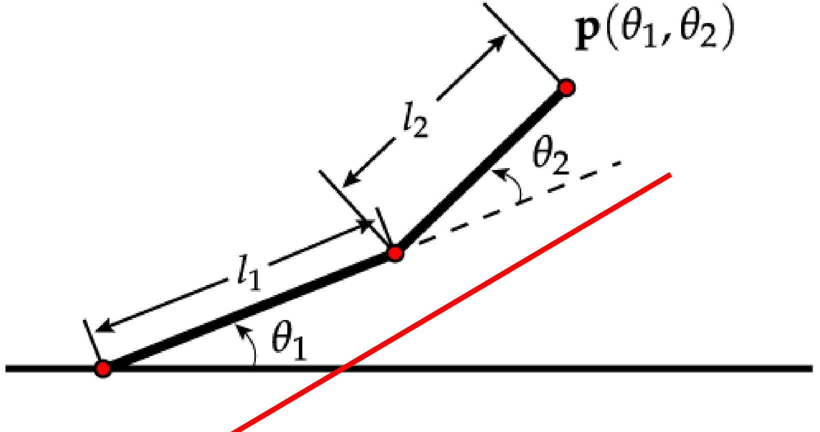
Helge Rhodin

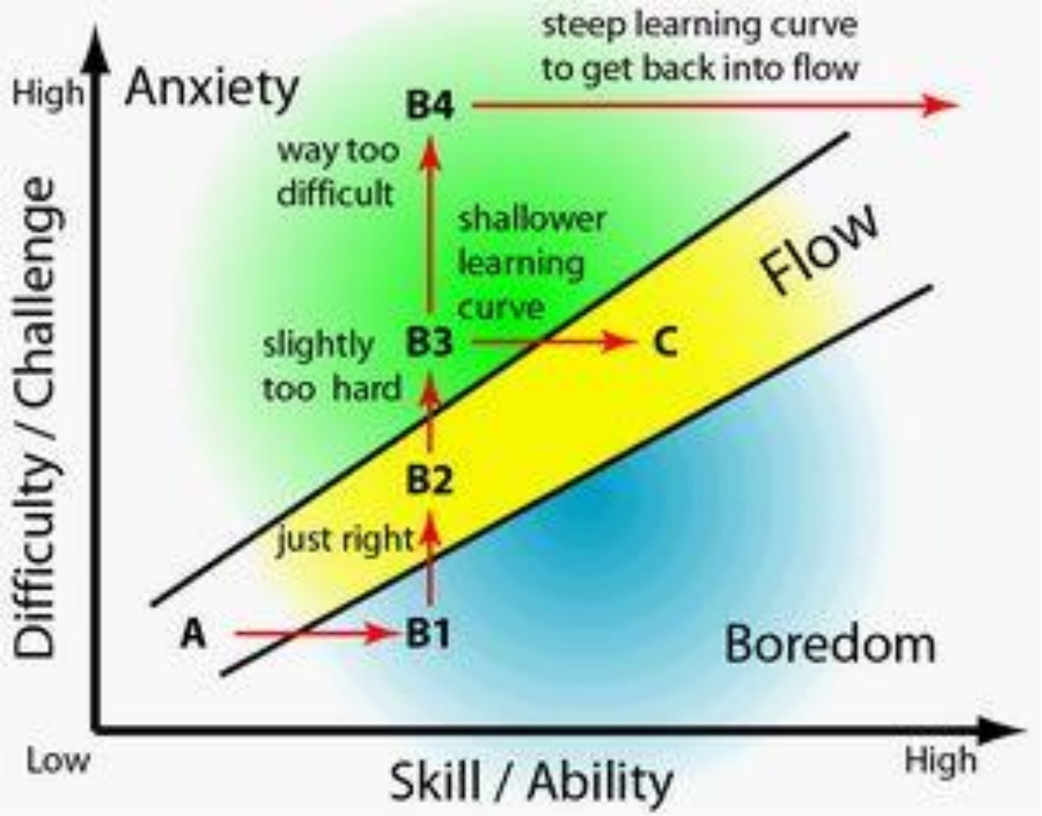
CPSC 427
Video Game Programming

Transformations for Skeleton Kinematics





Recap: Fun to play?

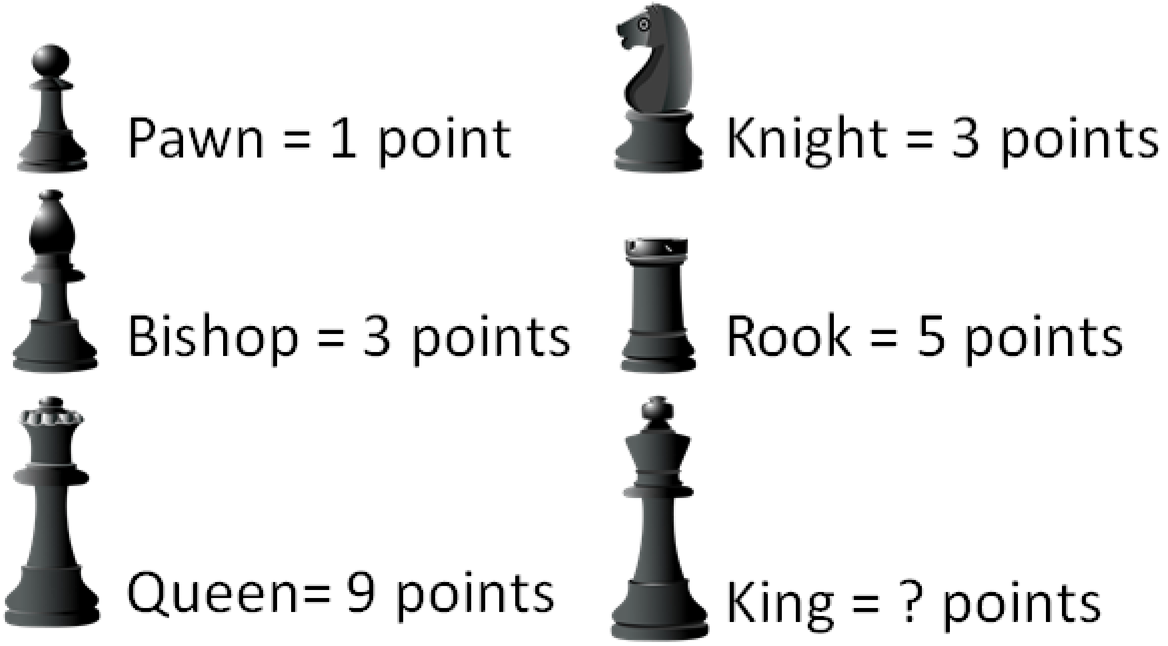


https://www.androidauthority.com/level-design-mobile-games-developers-make-games-fun-661877/

Recap: Indirect relationships

Value of a piece

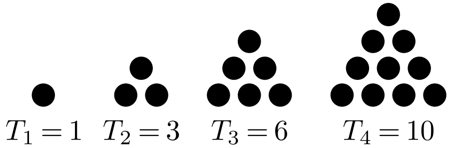
- *It is not possible to get a knight for 3 pawns*
- *But one can 'trade' pieces*
- *A currency*



How to determine the value?

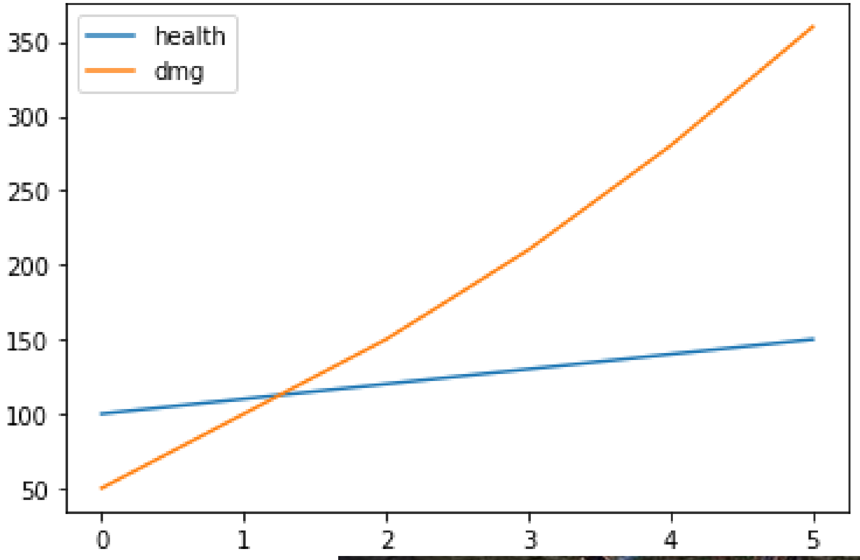
Recap: Relationships

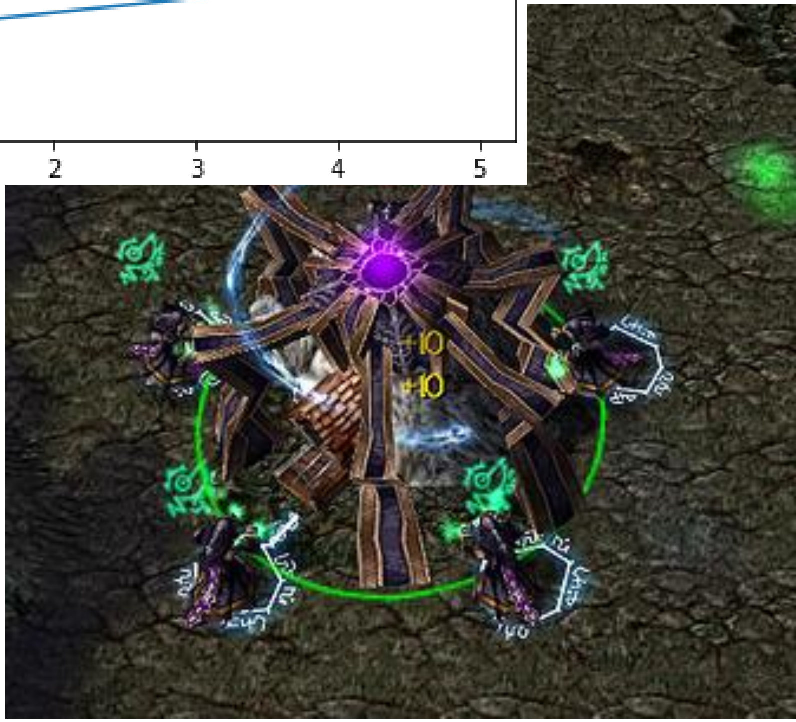
- ***Linear relations***
- ***Exponential relations***
- ***Triangular relationship***
  - 1, 3, 6, 10, 15, 21, 28, …



  - The difference increases linearly
  - The function has quadratic complexity

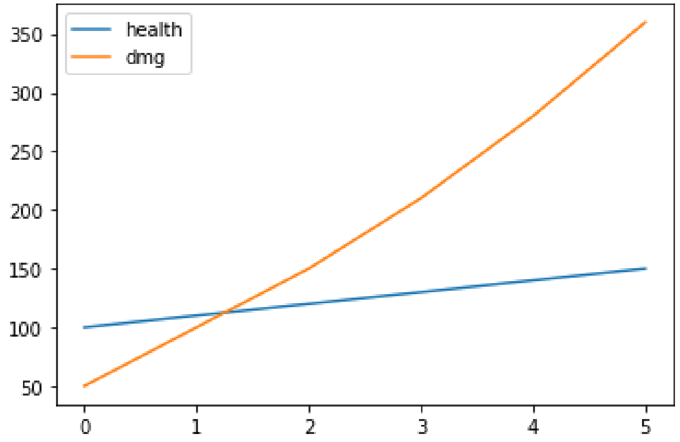
- ***Periodic relations***





Asymptotic analysis?

- *Linear * linear?*
- *Linear + linear?*
- *Linear + exponential?*
- *Linear * exponential?*



Formally, given functions $f(x)$ and $g(x)$, we define a binary relation

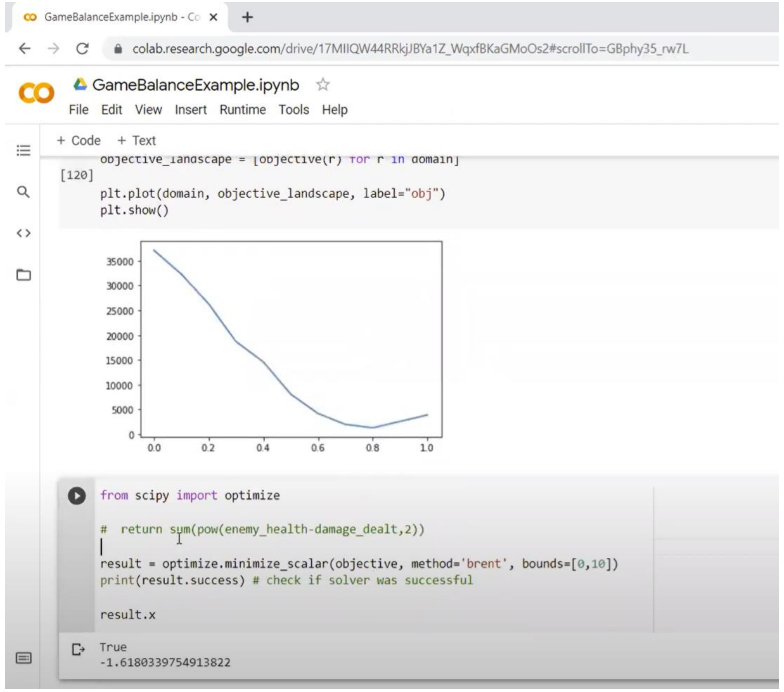
$$f(x) \sim g(x) \quad (\text{as } x \to \infty)$$

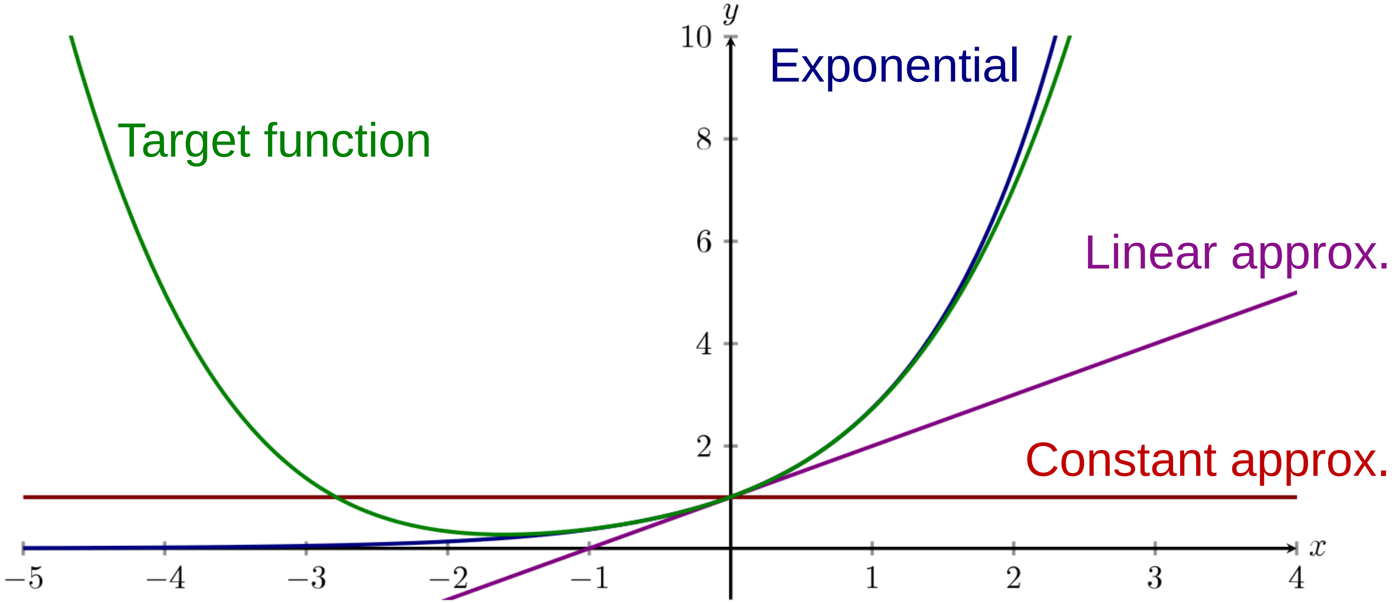
if and only if (de Bruijn 1981, §1.4)

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1.$$

Numerical Methods - Optimization

- ***Iterative optimizers***
  - Single variable?
  - Multiple variables?
  - Gradient descent?





Lecture 14: https://youtu.be/ZNsNZOnrM50

- **Balancing demo starts at 1h20**
- **Optimizer used at ~ 1h30**

Difficulties:

- ***Placement of towers changes the time damage is dealt***

- ***Path of enemies can be hindered to increase time***

➢ ***Measure during playtest***
  - ➢ *cross-play*

- ***Some enemies are resistant to fire/magic/…?***
  - *kind of a periodic feature*

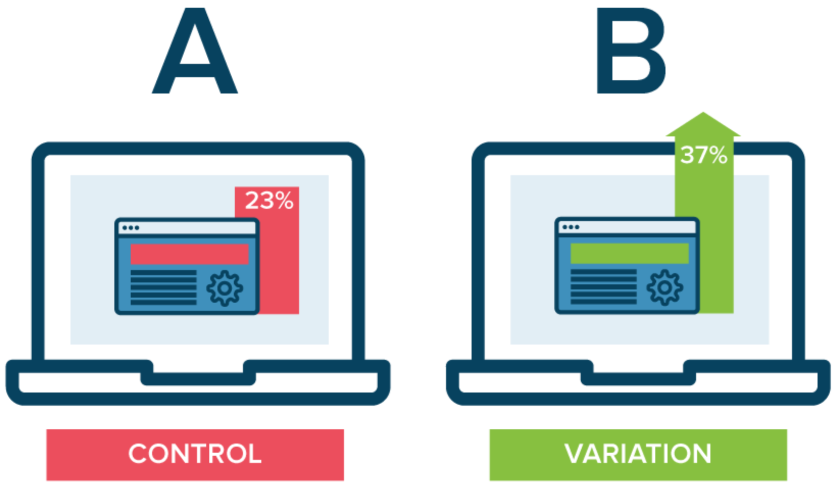
Counter Measures

- **Transitive Mechanics**
  - *Repair costs*
  - *Consumables (food, potions, …)*
  - *Tax*

A/B Testing

Testing two variants of your game (with and w/o a feature)

- *randomized participants (same pool)*
- *with respect to a measurable objective (e.g., clicks on website)*



Related to

- two-sample hypothesis testing
- Clinical tests, e.g., testing of a COVID-19 vaccine
- Placebo effect

https://www.optimizely.com/optimization-glossary/ab-testing/

Logistics

M4 updated requirement

~~***User study***~~

Carefully balance one aspect of your game (e.g., movement-speed, health points, strength, bonus,…).

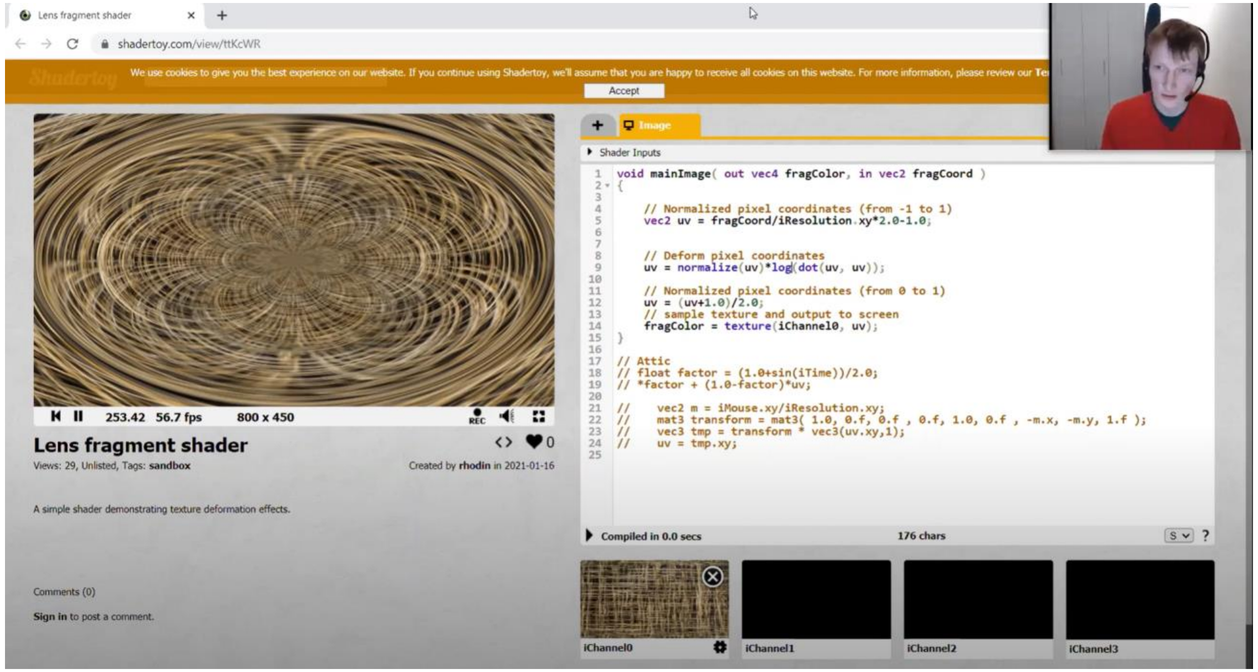
- ***Report on the theoretical analysis***
- ***Change log with testing***

M4 Advanced shaders?

- ***Use ShaderToy.com***

- ***Demo in Lecture 4***
  https://youtu.be/S97-fYMv4Xk

  45 minute mark



Upcoming lectures:

Tuesday:

- Working in teams
- Supported by TA

Thursday:

- *Last formal lecture:*
  - Skeleton Animation continued
  - Summary and Outlook

Tuesday (April 13.):

- *M4 grading with TAs*

Monday (April 19.):

- *Final cross play, including industry jury and awards*

Final cross play

Room	Purpose
1	Team 1 – open cross-play
2	Team 2 – open cross-play
…	…
12	Team 12 – open cross-play
13	Jury 1 (Skybox) – scheduled 10 min slots
14	Jury 2 (EA) – scheduled 10 min slots
…	…
20	Overflow – breakout for open cross-play
21	Overflow – breakout for open cross-play
…	…

Timeline:

7 – 8 pm: scheduled (10 minutes) cross play
7 – 8:30 pm: open cross play (those not with jury)
8 – 8:30 pm: jury votes, student votes are counted
8:30 – 9 pm: awards!

Communication:

On Slack to reach beyond zoom rooms

- OK to include jury?

Presence:

- Mandatory!

Today

Becoming an expert on transformation

- *Mental picture*
- *Math*
- *Practical examples*

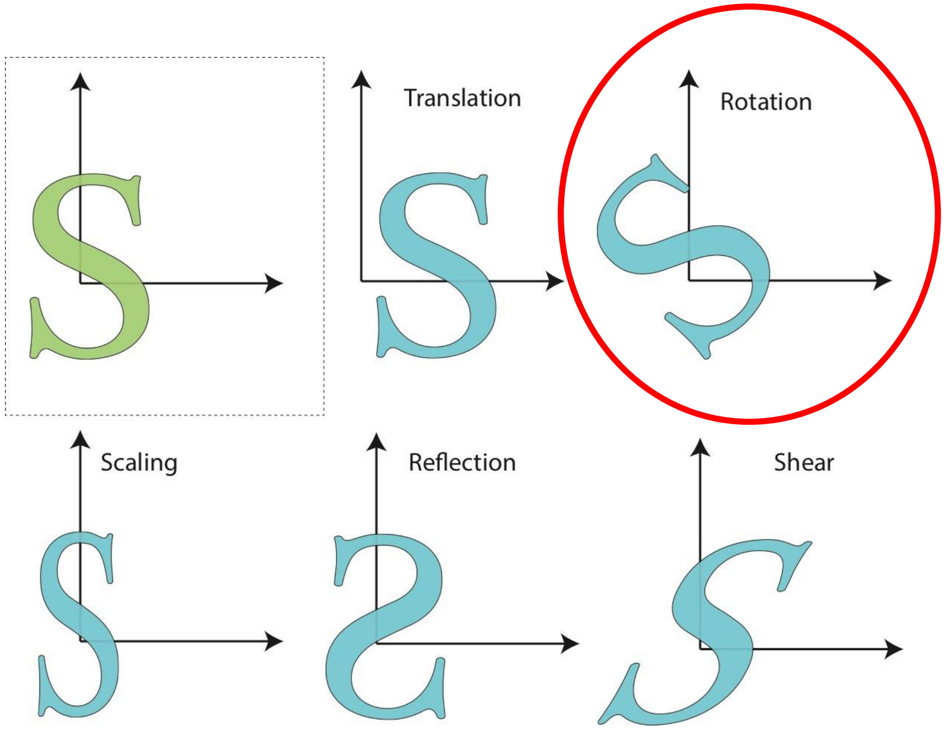
Composite transformations

- *Articulated skeleton motion*
- *Skeleton animation*

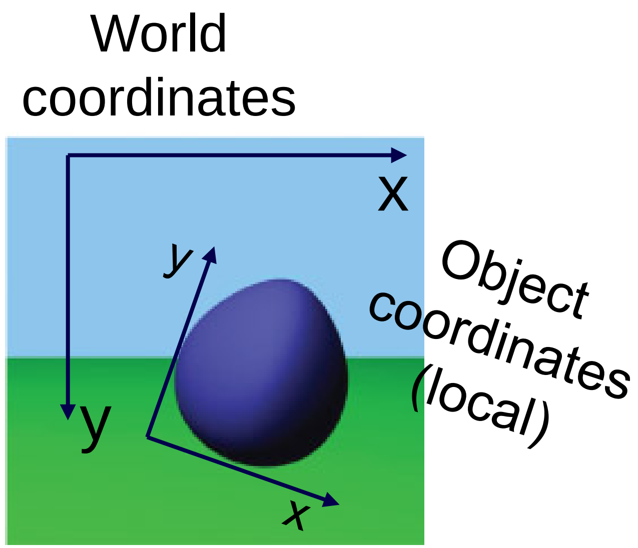
Recap: Transformations

Lecture 3, 20 minute mark

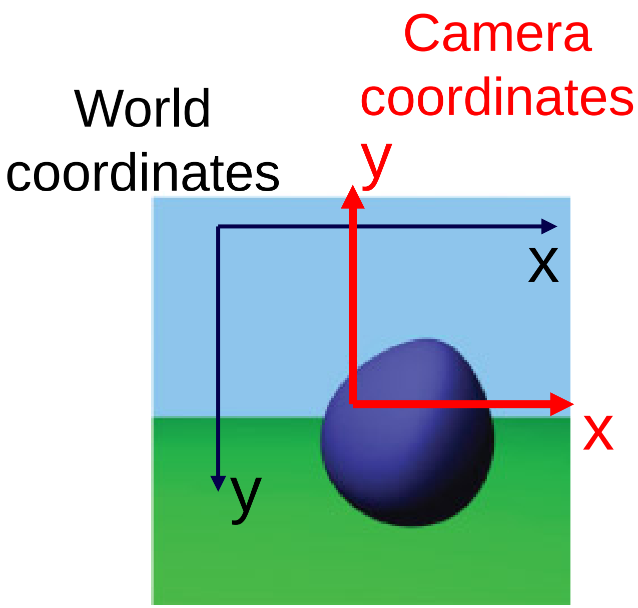
- *https://youtu.be/l9xQUxJT7fg*



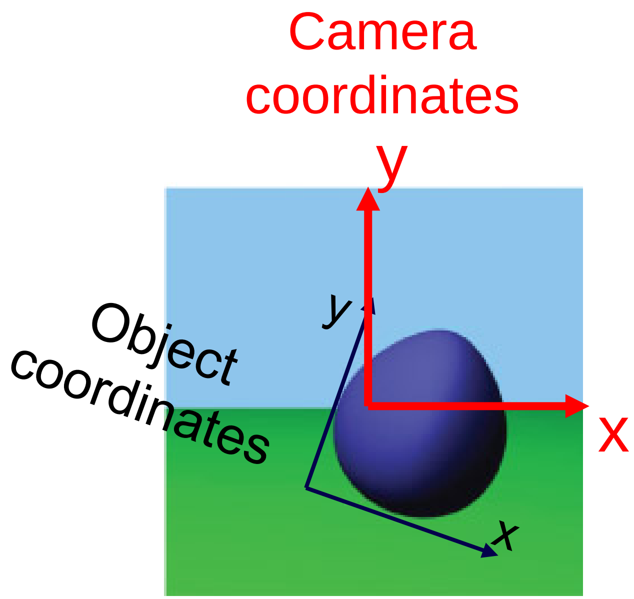
From local object to camera coordinates



object -> world



world -> camera



object -> camera

`transform` `projection` `projection * transform`

Recap: GLSL Vertex shader

The OpenGL Shading Language (GLSL)

- Syntax similar to the C programming language
- Build-in vector operations
  - functionality as the GLM library our assignment template uses

x and y coordinates of a vec2, vec3 or vec4

world -> camera

object -> world

```
void main()
{
    // Transforming The Vertex
    vec3 out_pos = projection * transform * vec3(in_pos.xy, 1.0);
    gl_Position = vec4(out_pos.xy, in_pos.z, 1.0);
}
```

vector of 3 (vec3) and 4 (vec4) floats

float (32 bit)

Affine transformations

- Linear transformations + translations
- Can be expressed as 2x2 matrix + 2 vector
- E.g. scale+ translation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Modeling Transformation

Adding a third coordinate

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \Rightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 2 & 0 & t_x \\ 0 & 2 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Affine transformations are now linear

- one 3x3 matrix can express: 2D rotation, scale, shear, and translation

Forward transformations

- ***Given position, scale, angle***
- ***Compute transformation matrix***
- ***Transform the object***

- ***Examples?***

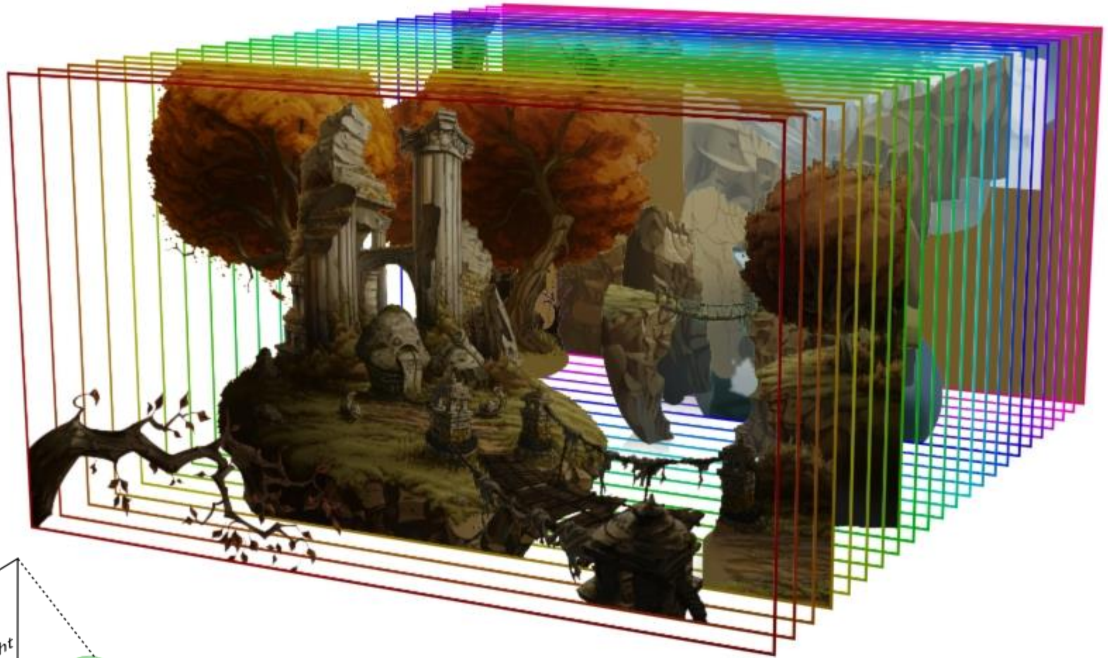
- ***Later: Inverse transformations***

Parallax scrolling background

Further objects

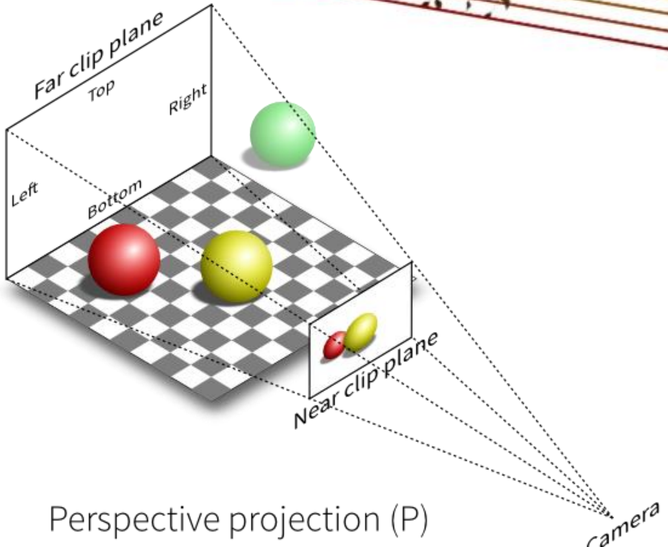
- ***are smaller***
- ***move slower***

Multiple layers!



Weak perspective projection

- *Approximates perspective projection*



Perspective projection (P)

Parallax scrolling background

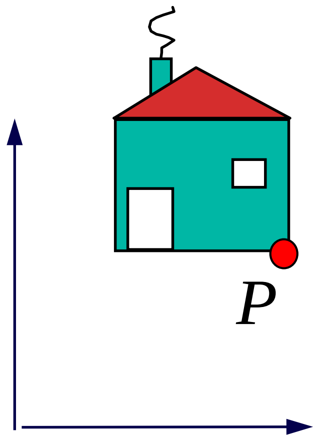
Ingredients:

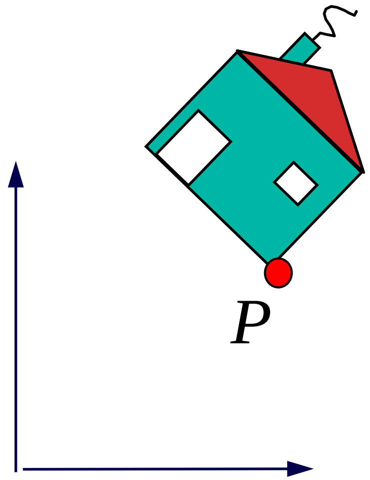
- *Multiple layers*
- *One sprite per layer*
- *One transformation per layer*
  - *Scale down with distance*
  - *Move inversely proportional to the distance*

$$\begin{pmatrix} 2 & 0 & t_x \\ 0 & 2 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

TRANSFORMATION COMPOSITION

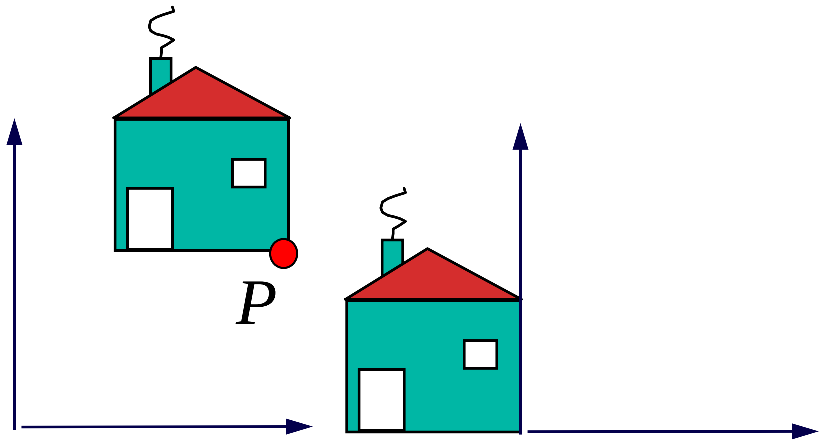
- **What operation rotates XY by θ around** $P = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$ **?**

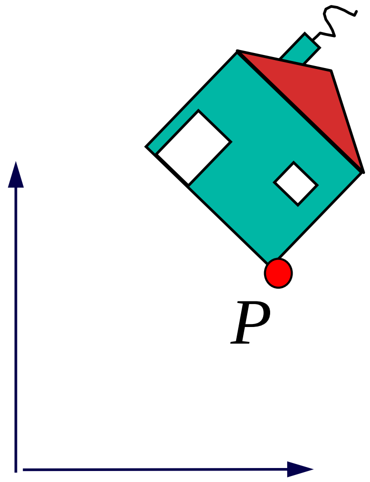




TRANSFORMATION COMPOSITION

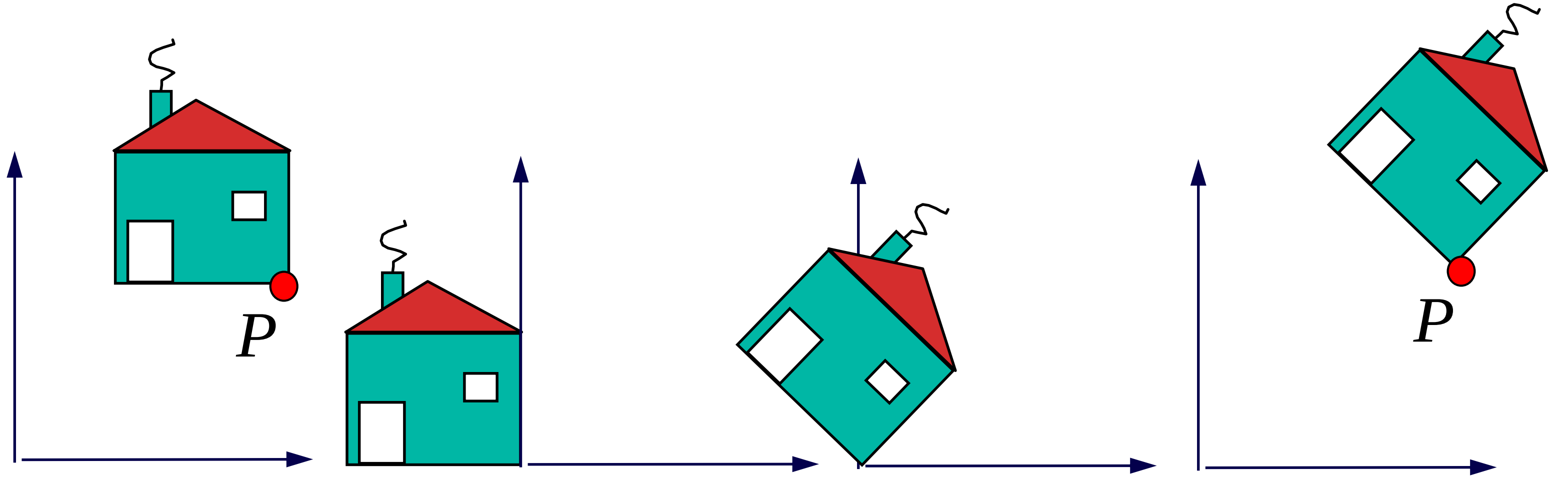
- **What operation rotates XY by θ around** $P = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$**?**
- **Answer:**
  - **Translate *P* to origin**





TRANSFORMATION COMPOSITION

- **What operation rotates XY by $\theta < 0$ around** $P = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$**?**
- **Answer:**
  - **Translate *P* to origin**
  - **Rotate around origin by θ**
  - **Translate back**



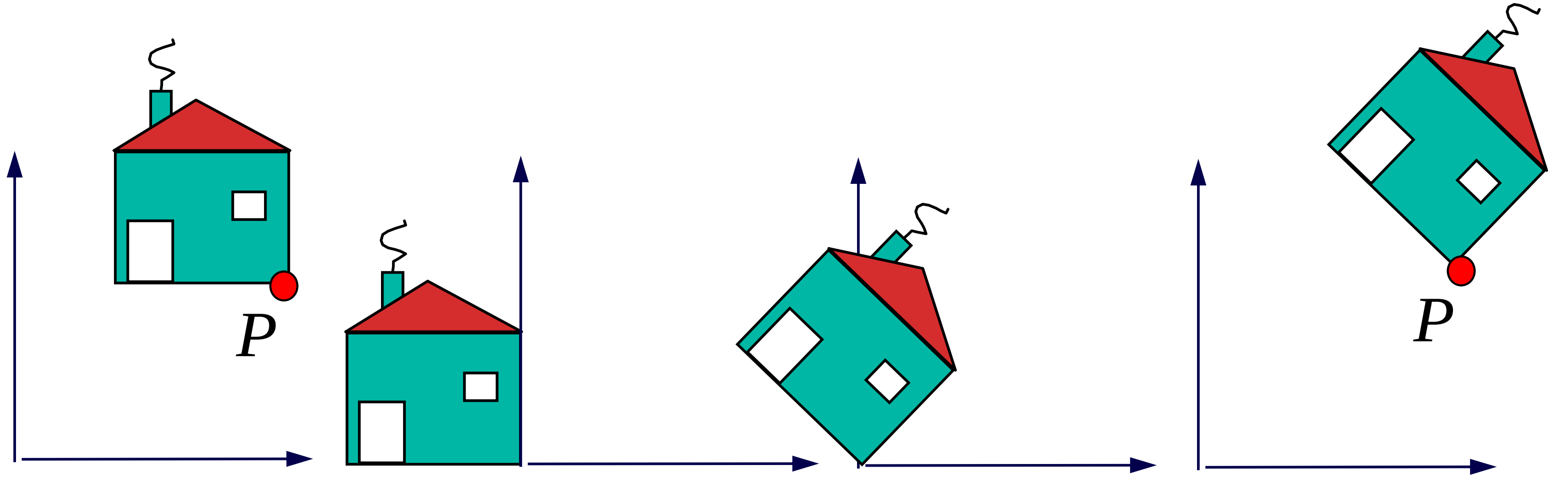
TRANSFORMATION COMPOSITION

$$T^{(p_x,p_y)}R^{\theta}T^{(-p_x,-p_y)}(V)$$

$$= \begin{bmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

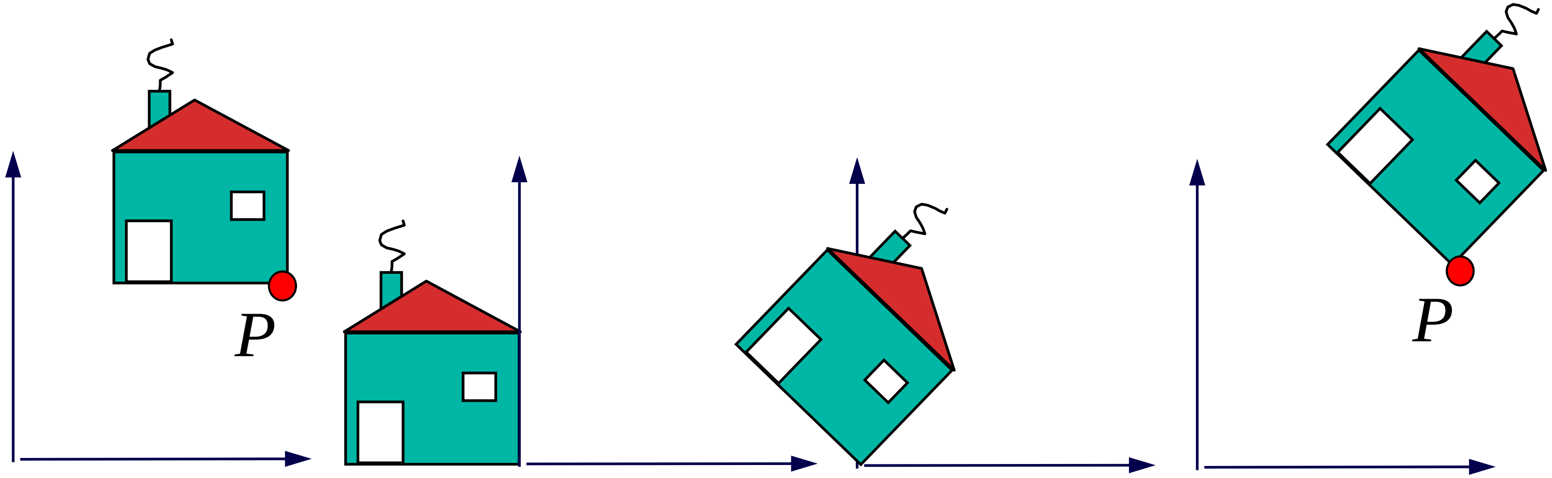
two interpretations

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



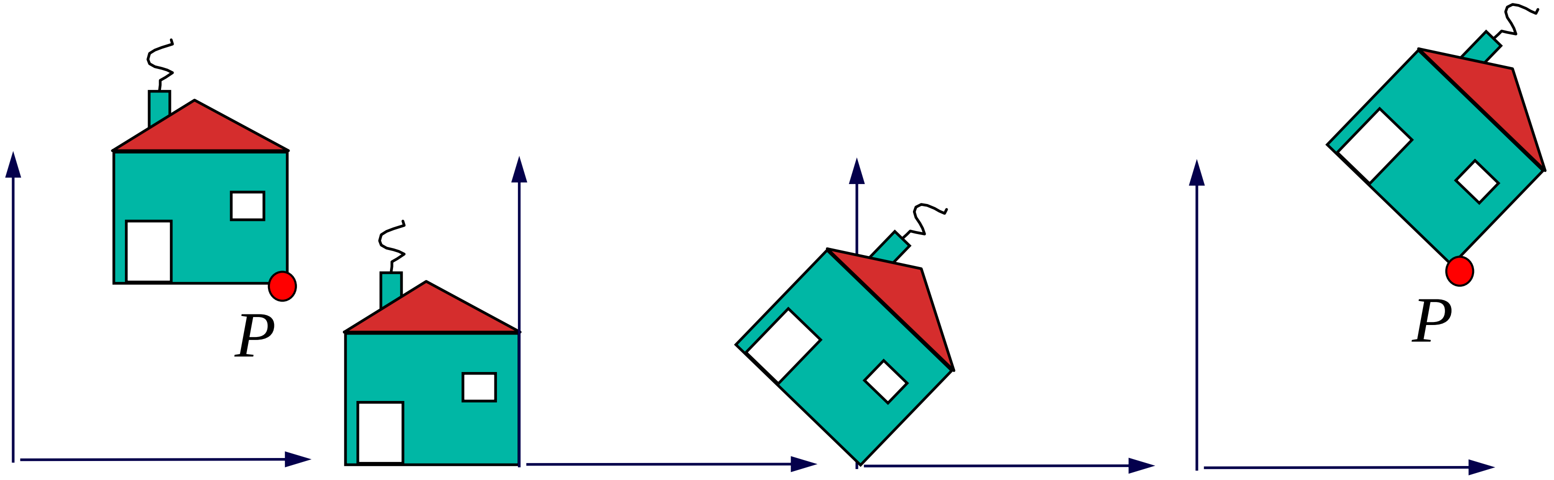
TRANSFORMING COORDINATES

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



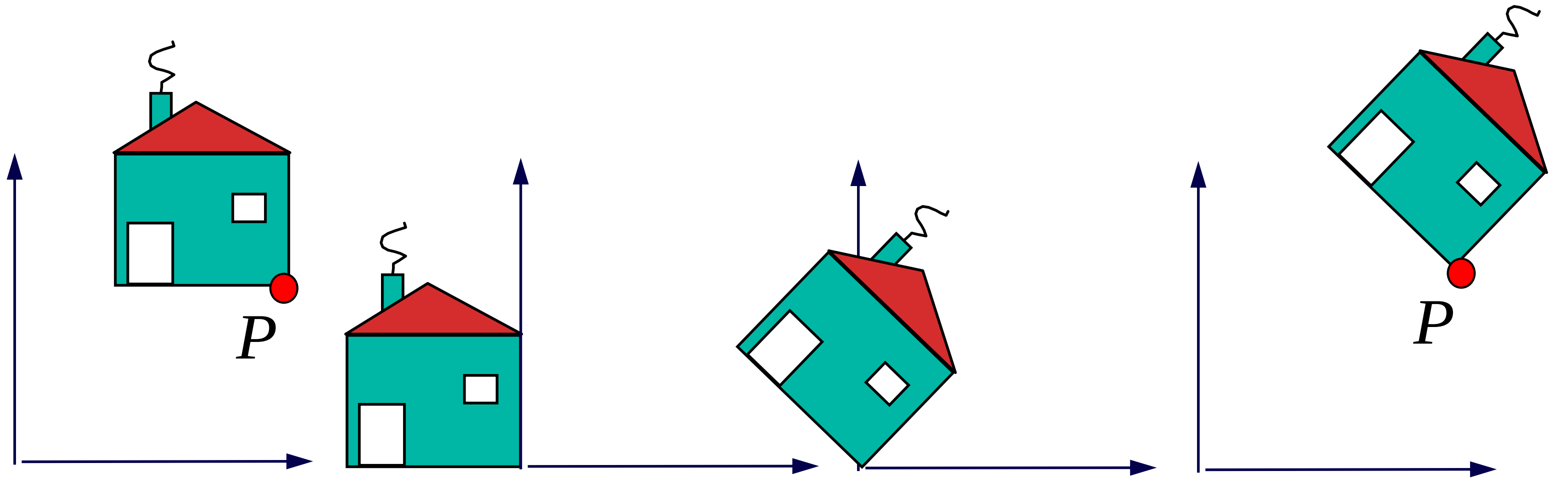
TRANSFORMING COORDINATES

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



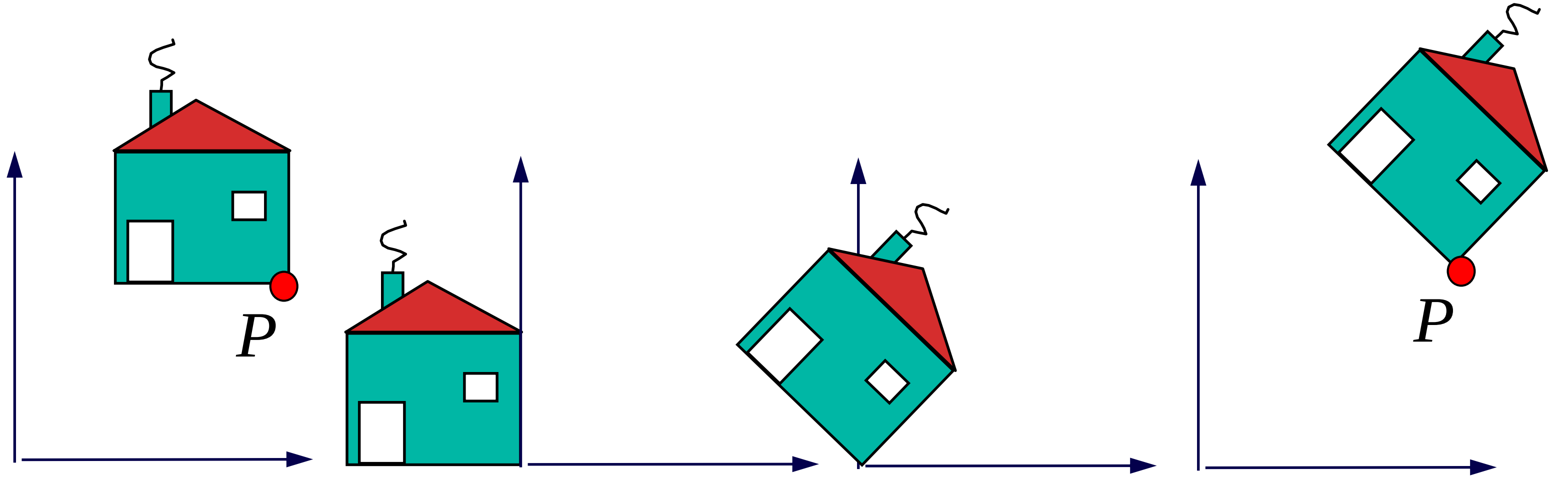
TRANSFORMING COORDINATES

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x\cdot(1-\cos\theta)+p_y\cdot\sin\theta \\ \sin\theta & \cos\theta & p_y\cdot(1-\cos\theta)+p_x\cdot\sin\theta \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



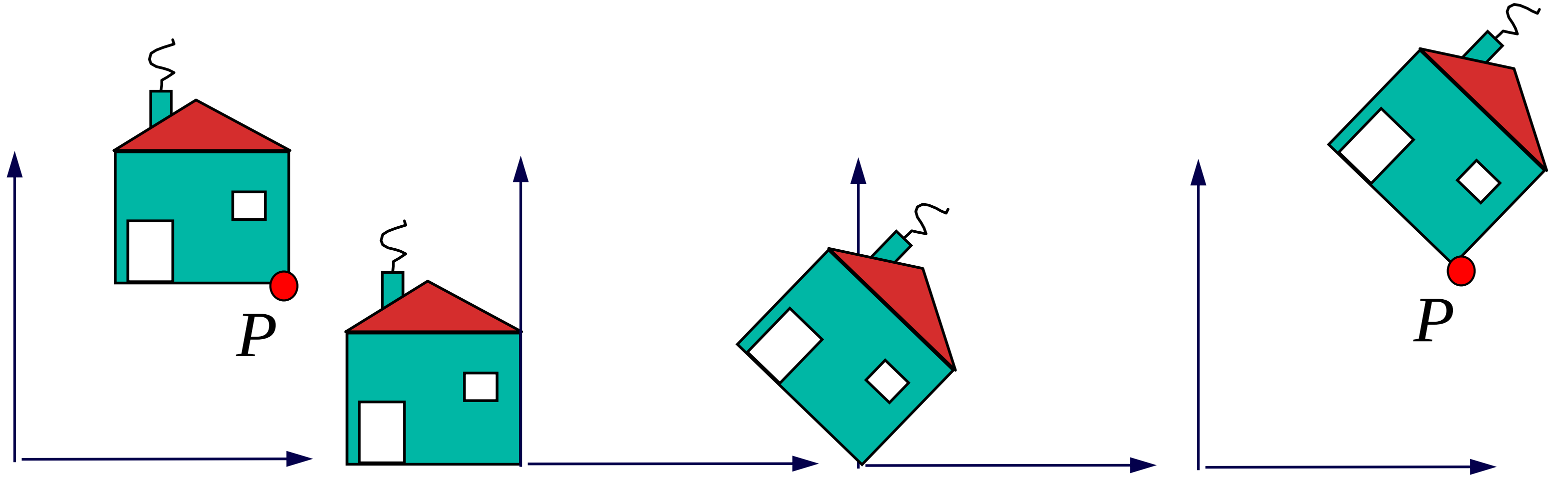
TRANSFORMING COORDINATE FRAME

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



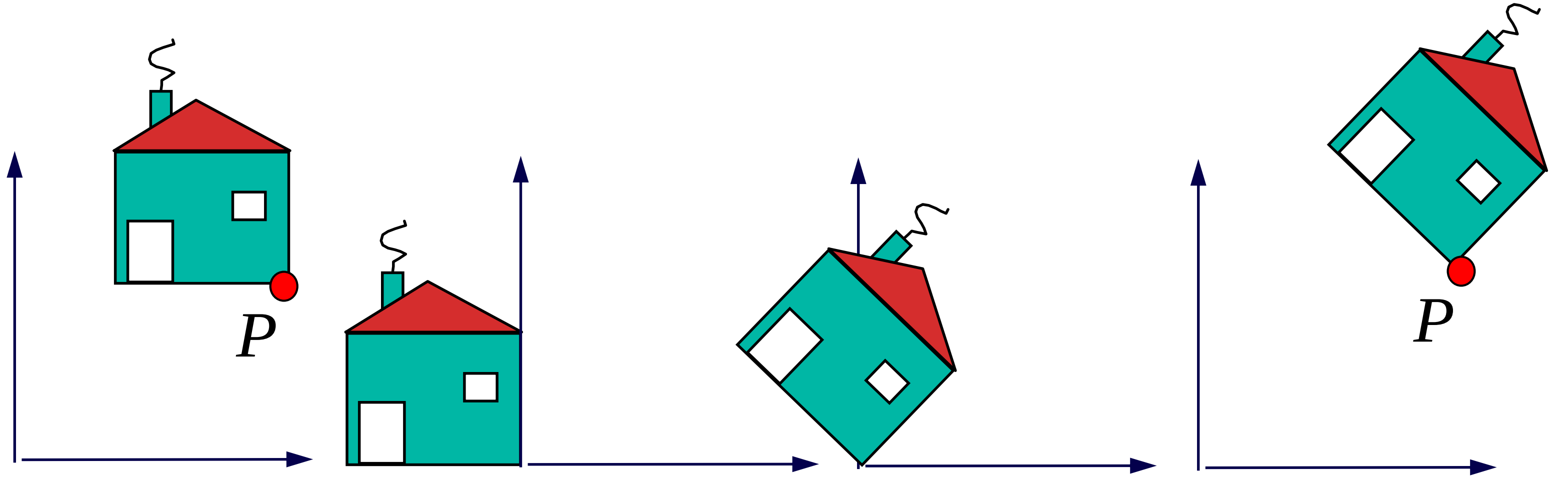
TRANSFORMING COORDINATE FRAME

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



TRANSFORMING COORDINATE FRAME

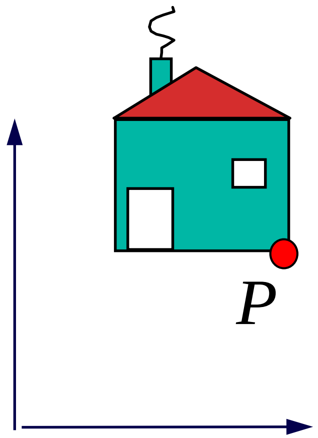
$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x \cdot (1-\cos\theta) + p_y \cdot \sin\theta \\ \sin\theta & \cos\theta & p_y \cdot (1-\cos\theta) + p_x \cdot \sin\theta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

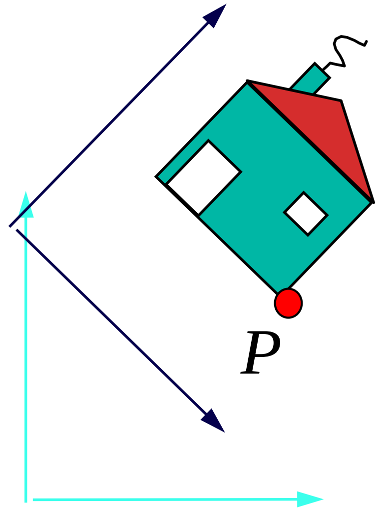


TRANSFORMING COORDINATE FRAME

Columns are new basis vectors (and new origin)!

$$\begin{pmatrix} \cos\theta & -\sin\theta & p_x\cdot(1-\cos\theta)+p_y\cdot\sin\theta \\ \sin\theta & \cos\theta & p_y\cdot(1-\cos\theta)+p_x\cdot\sin\theta \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



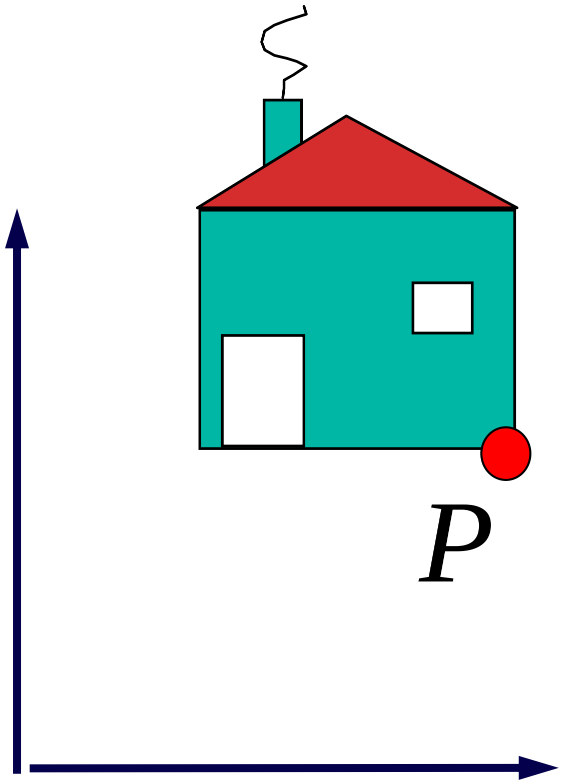


TRANSFORMING COORDINATE FRAME

$$T^{(p_x,p_y)} R^{\theta} T^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

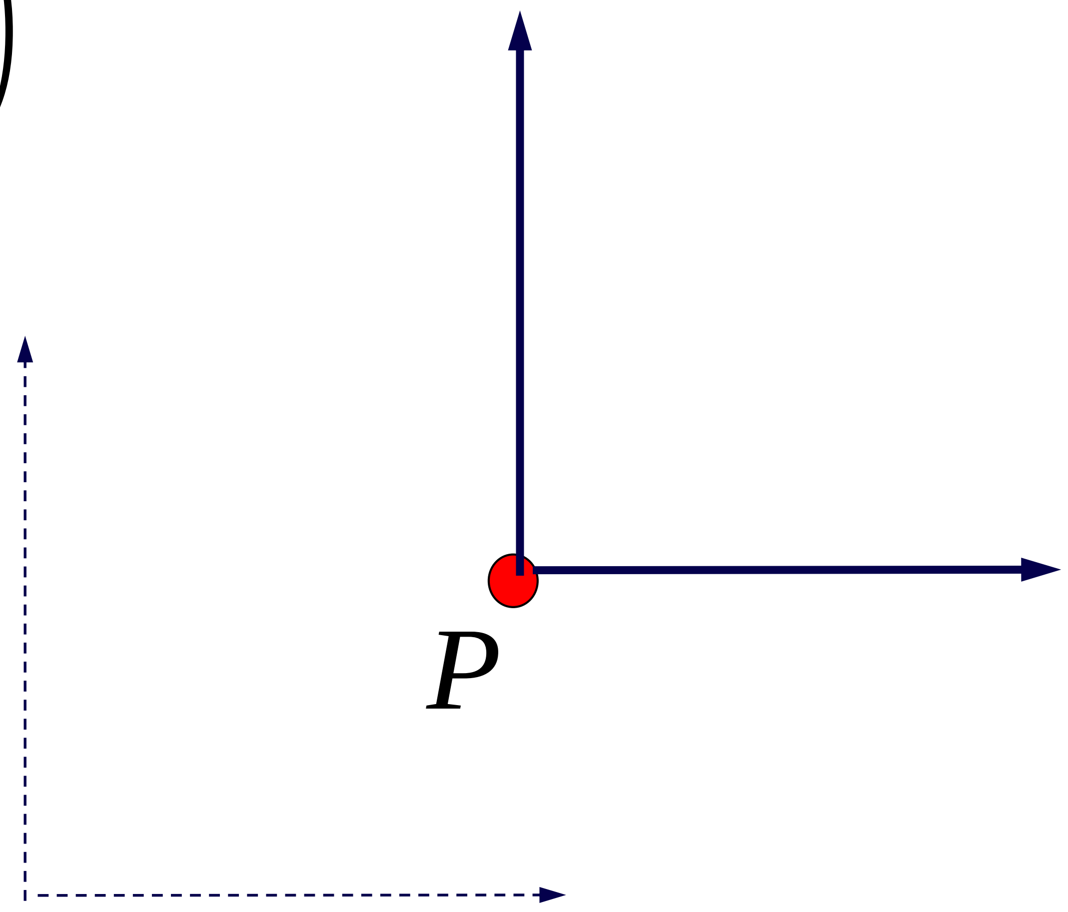
TRANSFORMING COORDINATE FRAME

$$\boldsymbol{T}^{(\boldsymbol{p}_x,\boldsymbol{p}_y)} R^{\theta} T^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



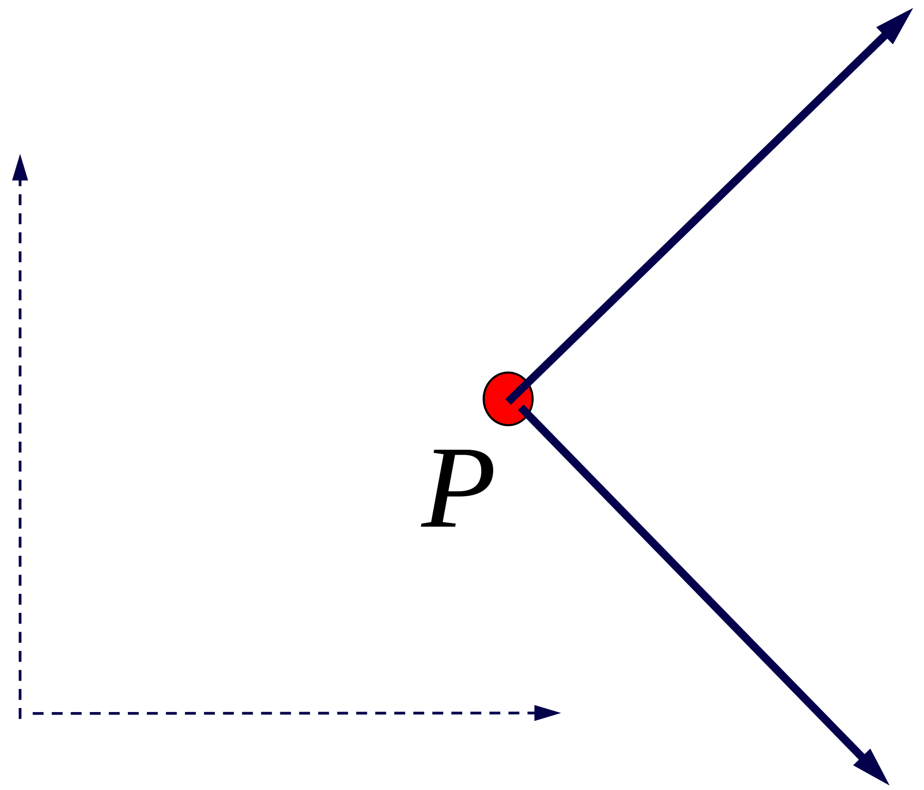
TRANSFORMING COORDINATE FRAME

$$\boldsymbol{T}^{(\boldsymbol{p}_x,\boldsymbol{p}_y)} R^{\theta} T^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



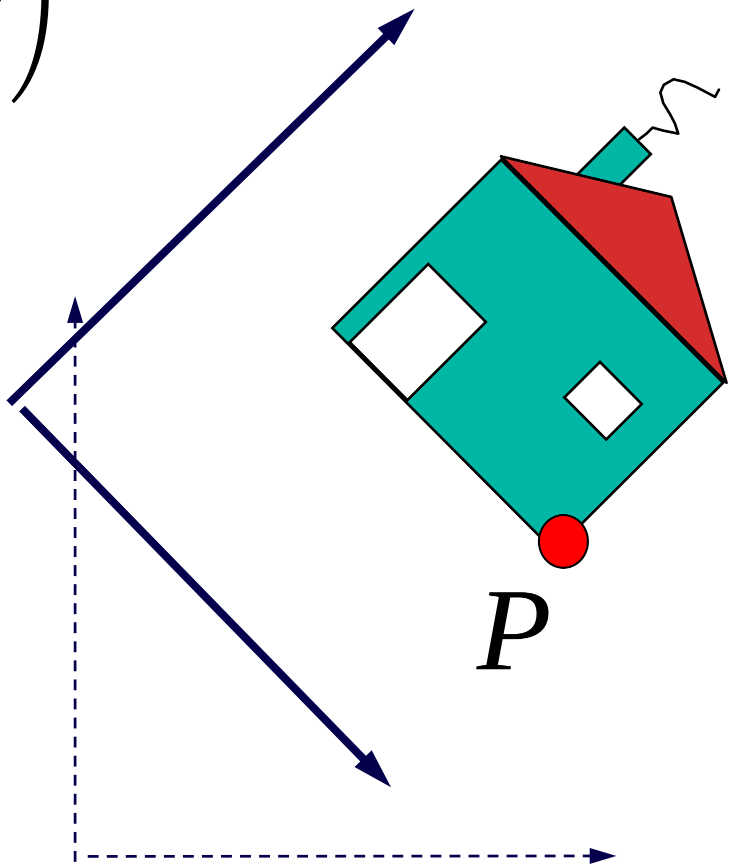
TRANSFORMING COORDINATE FRAME

$$T^{(p_x,p_y)}\boldsymbol{R}^{\boldsymbol{\theta}}T^{(-p_x,-p_y)}\begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



TRANSFORMING COORDINATE FRAME

$$T^{(p_x,p_y)} R^{\theta} \boldsymbol{T}^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

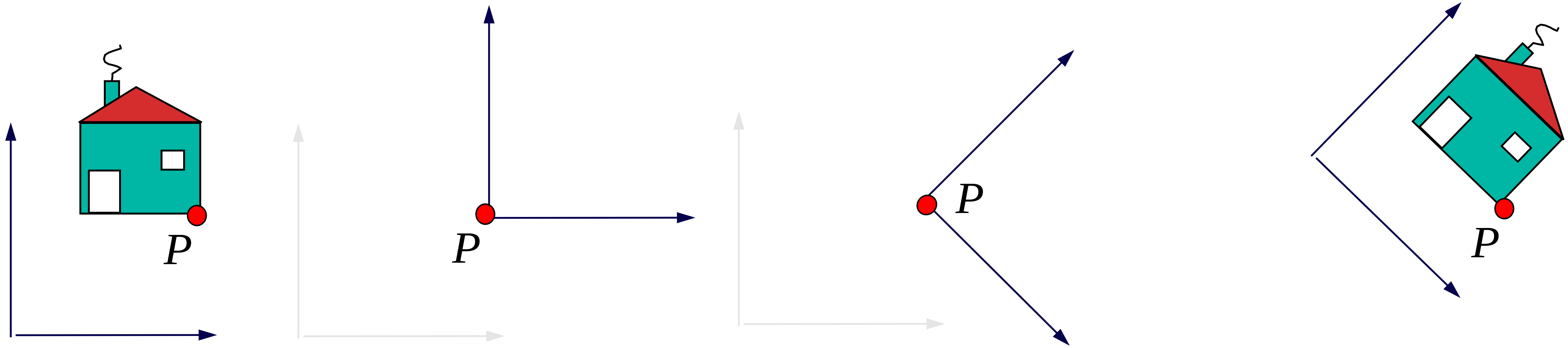


TRANSFORMING COORDINATE FRAME

World Coordinate Frame

Object Coordinate Frame

$$\begin{pmatrix} v'_x \\ v'_y \\ 1 \end{pmatrix} = T^{(p_x,p_y)} R^{\theta} T^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$



TWO INTERPRETATIONS OF COMPOSITE

World Coordinate Frame **Object Coordinate Frame**

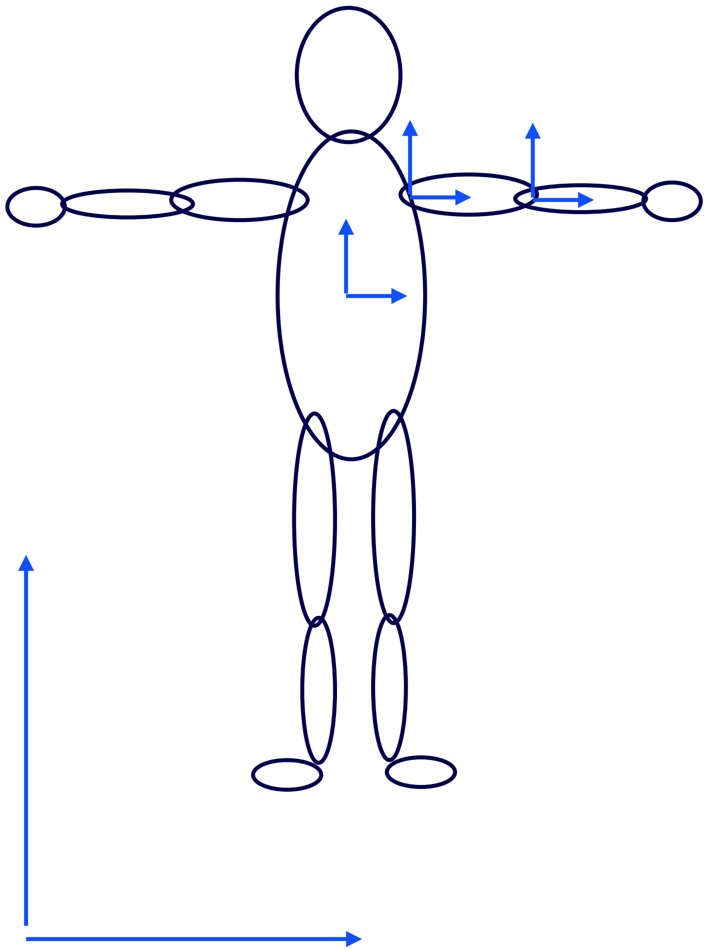
$$\begin{pmatrix} v'_x \\ v'_y \\ 1 \end{pmatrix} = T^{(p_x,p_y)} R^{\theta} T^{(-p_x,-p_y)} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$

1) read from inside-out as transformation of object

2) read from outside-in as transformation of the coordinate frame

Transformation Hierarchies

Transformation Hierarchies



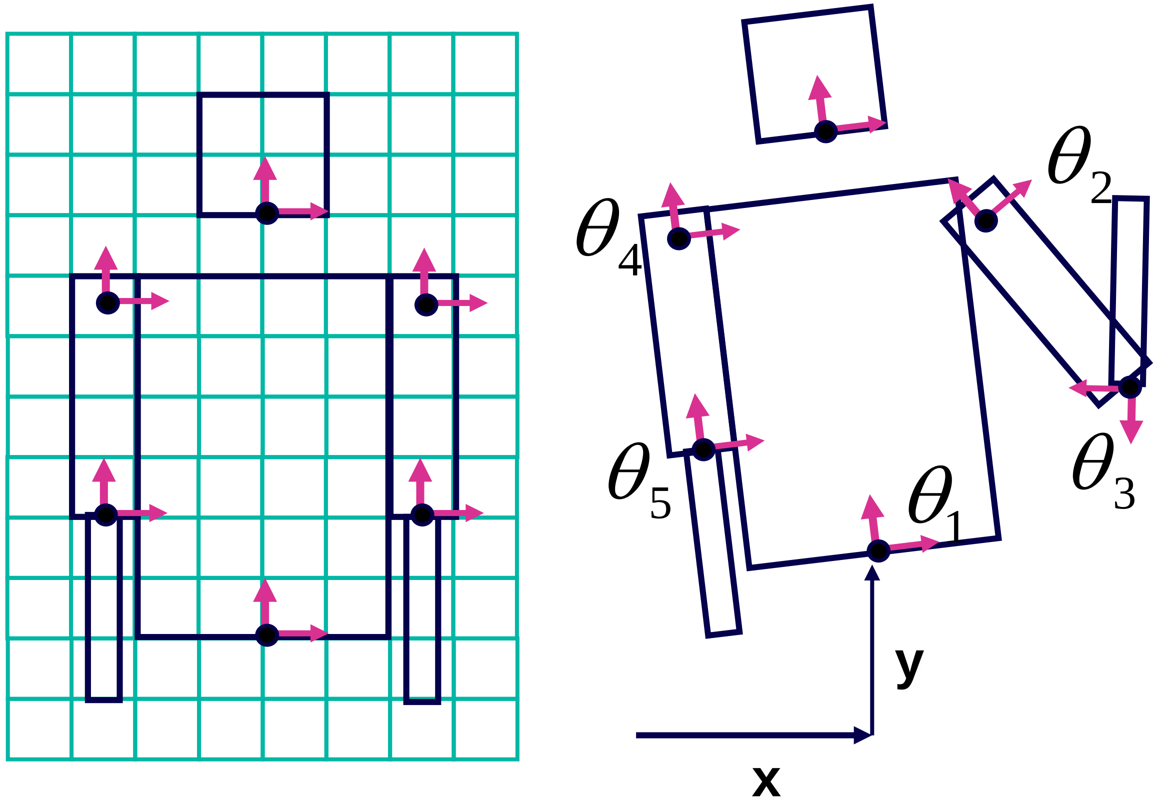
Scenes have multiple coordinate systems

- Often strongly related
  - *Parts of the body*
  - *Object on top of each other*
    - Next to each other…

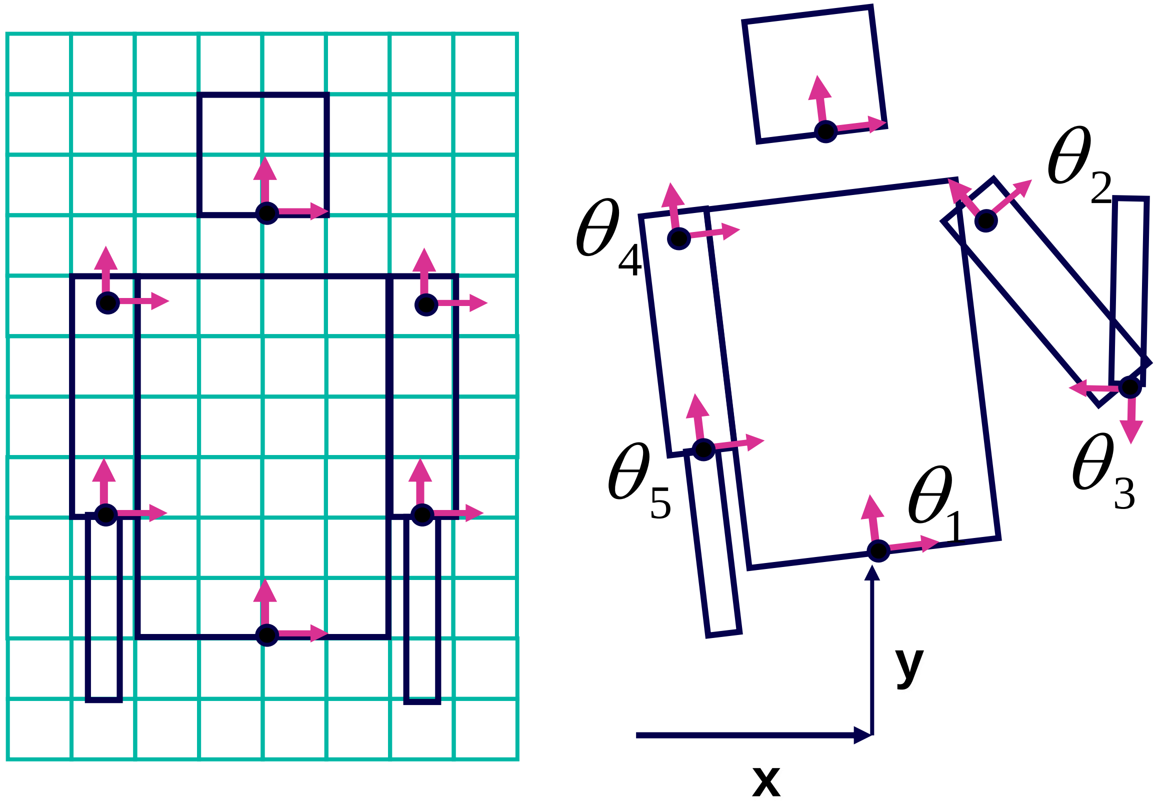
Independent definition is bug prone

Solution: Transformation Hierarchies

Transformation Hierarchy Examples



Transformation Hierarchy Examples

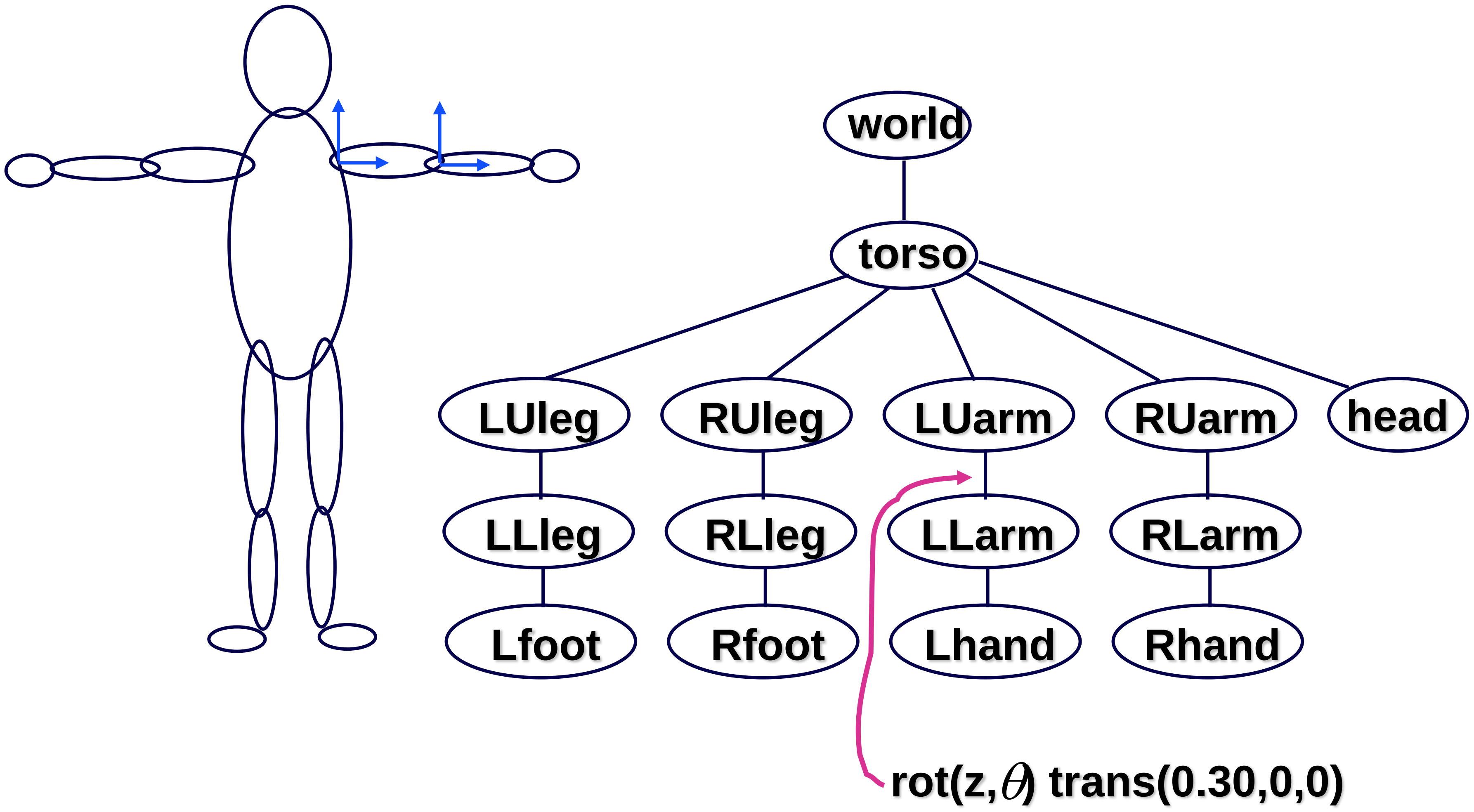


$$M_1 = Tr_{(x,y)} \cdot Rot\theta_1$$

$$M_2 = M_1 \cdot Tr_{(2.5,5.5)} \cdot Rot\ \theta_2$$

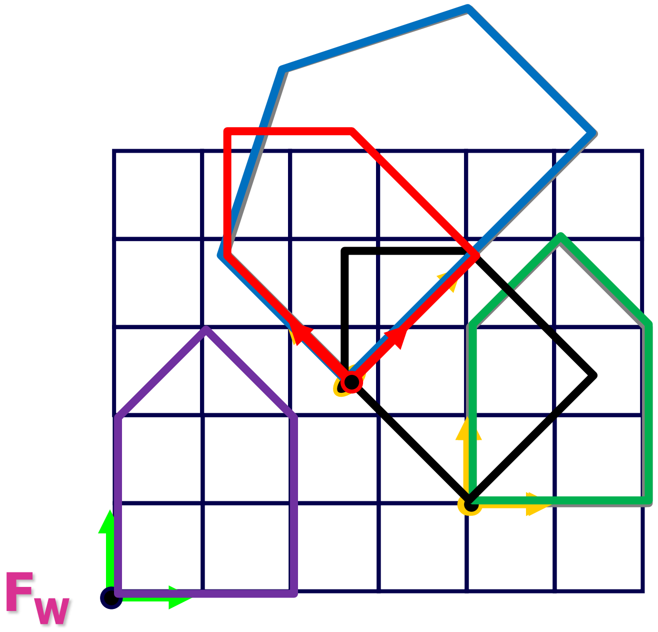
$$M_3 = M_2 \cdot Tr_{(0,-3.5)} \cdot Rot\ \theta_3$$

Transformation Hierarchies



Transformation Hierarchy Quiz

```
M.setIdentity();
M = M*Translation(4,1,0);
M = M*Rotation(pi/4,0,0,1);
House.matrix = M;
```



Which color house will we draw?

A. Red
B. Blue
C. Green
D. Orange
E. Purple

Hierarchical Modeling

Advantages

- Define object once, instantiate multiple copies
- Transformation parameters often good control knobs
- Maintain structural constraints if well-designed

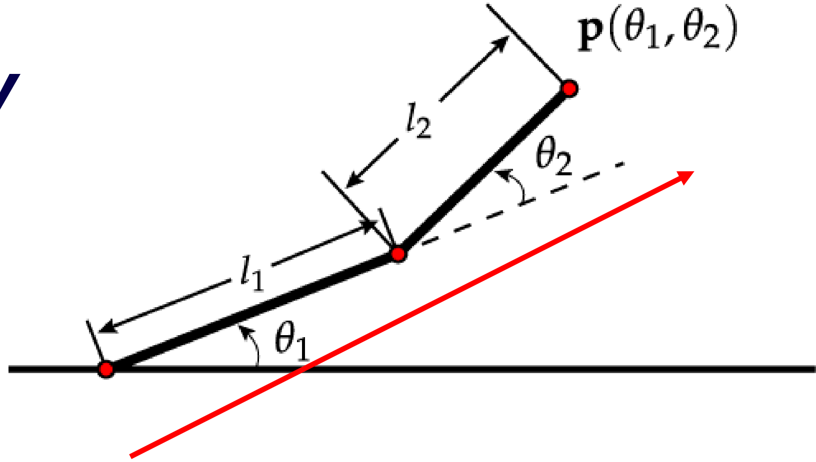
Limitations

- Expressivity: not always the best controls
- Can’t do closed kinematic chains
  - *E.g., how to keep a hand on the hip?*

Forward vs. inverse kinematics

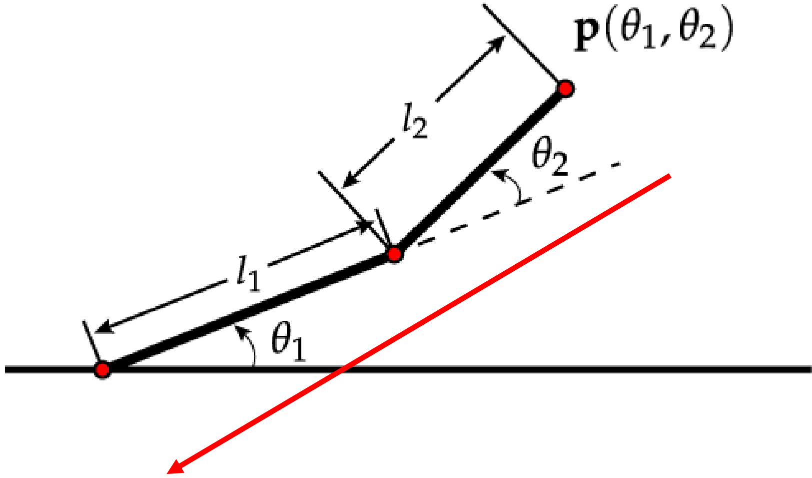
Forward kinematics

- ***given joint axis, angle, and skeleton hierarchy***
- ***compute joint locations***
  - *start at the end-effector (e.g. arm)*
    - rotate all parent joints (up the hierarchy) by θ
  - *iteratively continue from child to parent*



Inverse kinematics

- given skeleton hierarchy and goal location
- optimize joint angles (e.g. gradient descent)
- minimize distance between end effector (computed by forward kinematics) and goal locations



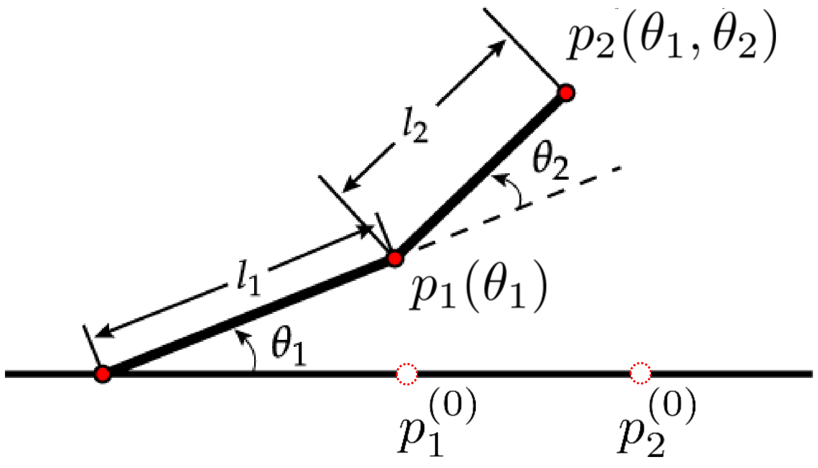
Inverse kinematics (IK)

- ***non-linear in the angle (due to cos and sin)***

$$M_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \end{bmatrix} \qquad M_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & -l_1 \\ \sin\theta_2 & \cos\theta_2 & 0 \end{bmatrix}$$

- linear/affine given a set of rotation matrices

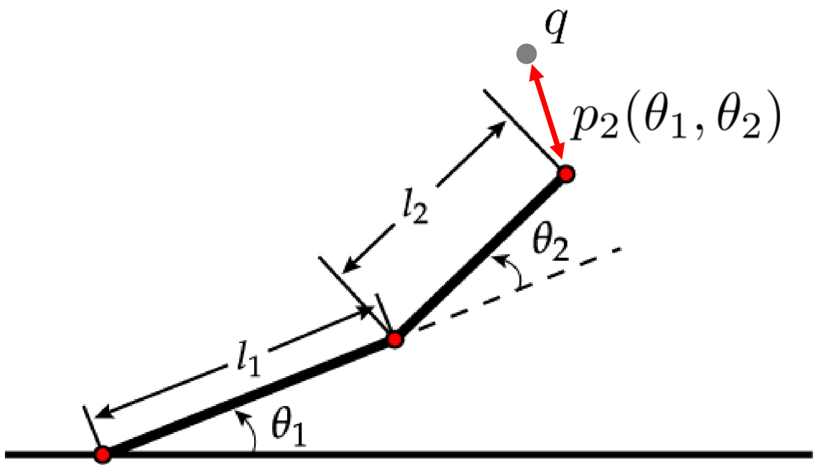
$$p_2(\theta_1, \theta_2) = M_1 M_2 (p_2^{(0)} - p_1^{(0)})$$



Inverse kinematics

- minimize objective to reach goal location

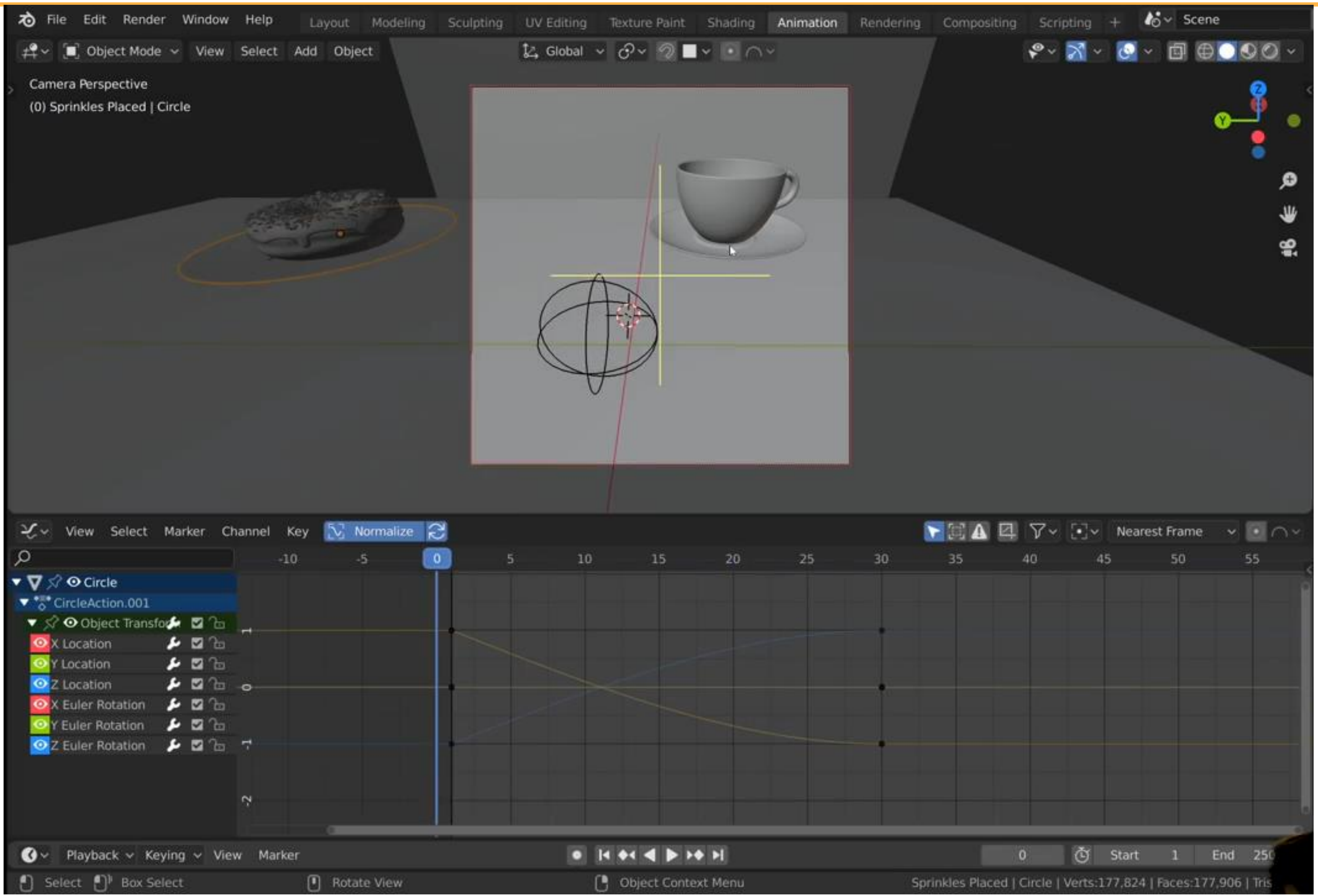
$$O(\theta_1, \theta_2) = \|q - p_2(\theta_1, \theta_2)\|$$



Example IK framework:

https://rgl.s3.eu-central-1.amazonaws.com/media/pages/hw4/CS328_-_Homework_4_3.ipynb

Recap:
Keyframe animation & mesh creation



Smooth curve