

2D vs. 3D

Squirrel Eiserloh

Director, TrueThought

Squirrel@TrueThought.com

Fight!

2D vs. 3D

Squirrel Eiserloh

Director, TrueThought

Squirrel@TrueThought.com

Relevance

Lots of people are using 2D

- Indie soloists
- Indie studios
- Casual game developers
- AAA game developers

Relevance



Braid



Gish



Viewtiful Joe



Odin Sphere

Relevance



Alien Hominid



Gradius

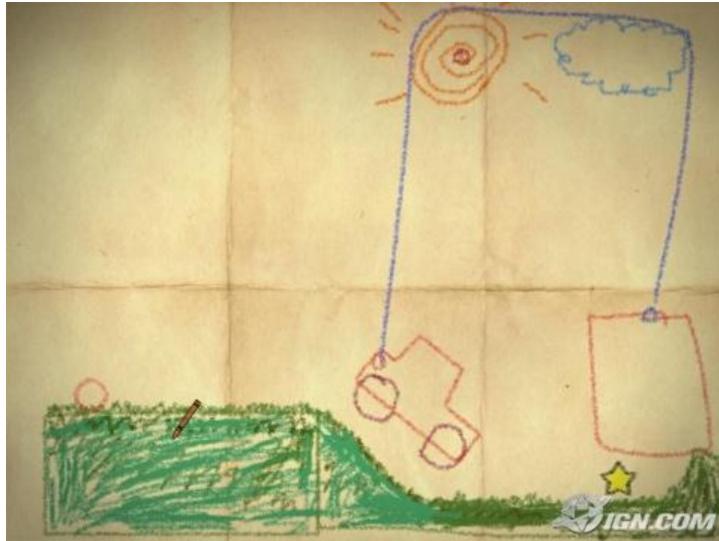


The Red Star

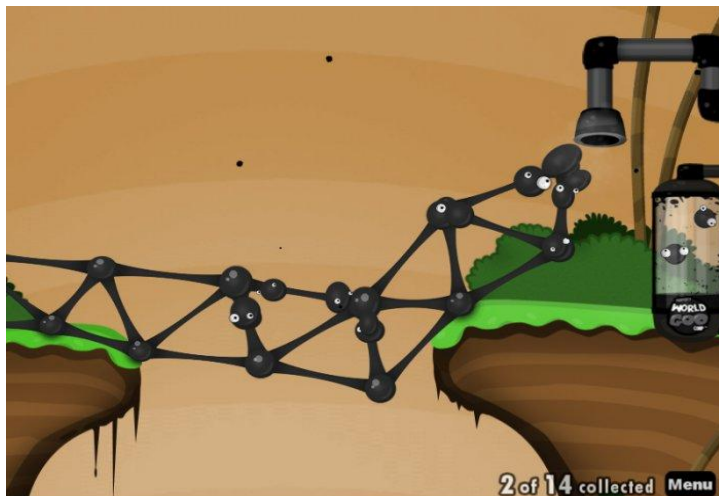


Contra

Relevance



Crayon Physics Deluxe



World of Goo



Armadillo Run



Fantastic Contraption

Motivation

2D Physics $\neq$ 3D Physics

sometimes $==$

sometimes $\neq$

Whitepapers & books often assume 3D

What is 2D, anyway?

Doesn't use OpenGL / Direct3D?

No, most 2D games use hardware accelerated "3D" graphics



What is 2D, anyway?

Doesn't use OpenGL / Direct3D?

No, most 2D games use hardware accelerated "3D" graphics (even simple ones)

The screenshot displays a SquareLogic puzzle game interface. On the left, a tutorial panel titled "Solving Addition" provides instructions on how to solve a cage. The main area features a 4x4 grid of puzzle cages. The top row contains four cages: a green cage with "2=" and the number "2", a yellow cage with "5+" and the number "3", a blue cage with "4=" and the number "4", and a green cage with "1=" and the number "1". The second row contains a blue cage with "12x" and numbers "1 3 4", a yellow cage with a red arrow pointing to a red-bordered cell containing the number "2" (with "1" and "4" also visible), and a purple cage with "36x" and numbers "1 2 3". The third row contains a blue cage with numbers "1 3 4", a purple cage with numbers "1 2 4", a purple cage with numbers "1 2 3", and a yellow cage with "Even" and numbers "2 3 4". The bottom row contains a blue cage with numbers "1 3 4", a brown cage with "3+" and numbers "1 2 4", a brown cage with numbers "1 2 3", and a yellow cage with numbers "2 3 4". At the bottom, a yellow button labeled "Click this button to go to the next step of the tutorial..." is visible.

Exit Tutorial

Restart Tutorial

Solving Addition

With the 3 in place, we can now solve the other square in this 5+ cage.

Since $3+2=5$, this square must be a 2.

In a real puzzle, SquareLogic shows you all of the possible equations for the current cage on the left side of the screen.

Solutions will turn RED when they are no longer possible (given the values you have set).

NOTE: In the options menu you can set equations to be hidden if you prefer.

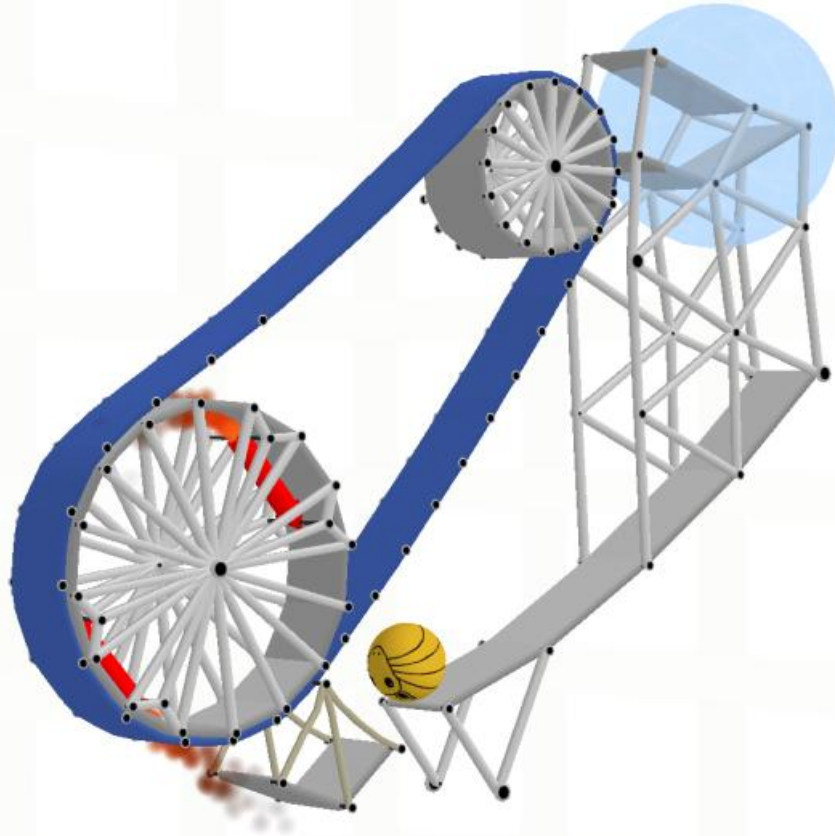
Next Page

Click this button to go to the next step of the tutorial...

What is 2D, anyway?

Uses 2D sprites rather than 3D models?

No, many 2D games use 3D models



What is 2D, anyway?

Things moving in two dimensions?

No, Luxor3 is actually 1D physics



What is 2D, anyway?

Uses non-perspective transform?



400
AMMO

73%
HEALTH

2 3 4
5 6 7
ARMS



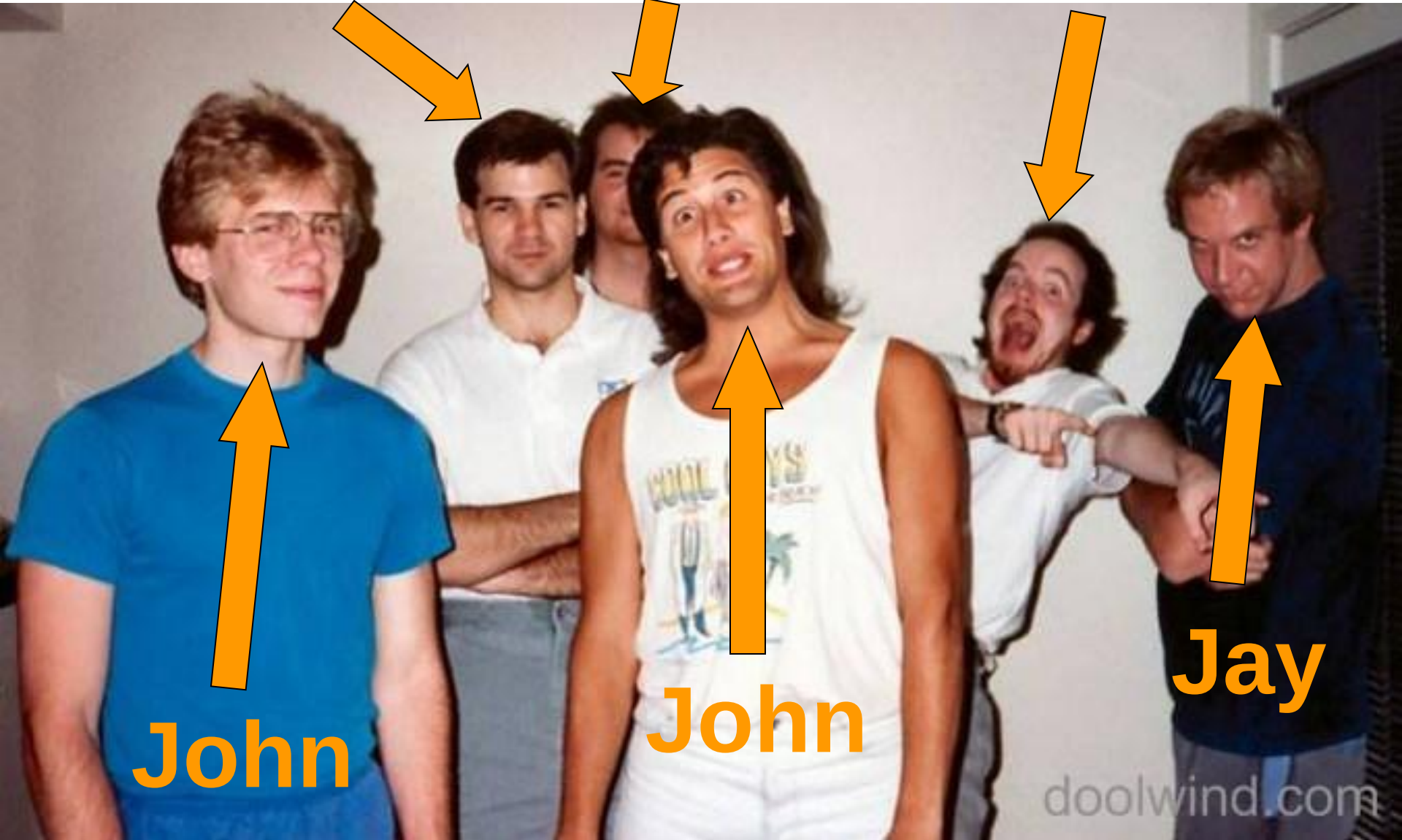
87%
ARMOR

BULL	400	/	400
SHEL	90	/	100
ROKT	38	/	100
CELL	40	/	600

Kevin

Adrian

Tom



Jay

John

John

id Software / 1993



5

AMMO

103%

HEALTH

1

FRAG



0%

ARMOR



BULL
SHEL
ROKT
CELL

50
0
5
0

200
50
50
300

Choose the object to edit:

- (1) Edit the linedef
- (2) Edit the 1st sidedef
- (3) A

Edit sidedef #182

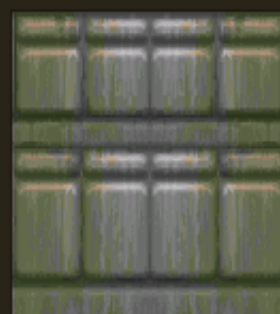
- (1) Change middle texture (Current: STARG3)
- (2) C
- (3) C
- (4) C
- (5) C
- (6) C

Choose a wall texture

STARG3

128x128

STARG3
STARGR1
STARGR2
STARTAN1
STARTAN2
STARTAN3
STEP1
STEP2
STEP3
STEP4
STEP5
STEP6
STEPLAD1
STEPTOP
STONE
STONE2
STONE3
STONGARG



Linedef #139

Flags: — — — — I
Type: 0 — Normal
Tag: 0 (none)
Vertices: (#125, #126)
Length: 256
1st sd: #182
2nd sd: #-1

Sidedef1 #182

Upper: —
Middle: STARG3
Lower: —
X-ofs: 0
Y-ofs: 0
Sector: #20

(no 2nd sidedef)

What is 2D, anyway?

- Doesn't use OpenGL / Direct3D?
No, many 2d games use hardware accelerated graphics
- Uses 2d sprites rather than 3d models?
No, platformers with 3d models are 2d games
- Things moving in two dimensions?
No, Luxor3 is actually 1D physics
- Uses non-perspective transform?
No, Doom used perspective transform in 2D world

So what is it?

2D Math & 2D Data

For our purposes here today, you may be doing some flavor of 2D if:

- Your data is 2D (x,y) or...
- You are doing 2D math

The Case for 2D

- Easier to diagram
- Easier to understand
- Easier to implement
- Easier to debug
- Easier to learn new algorithms
- Faster to compute
- Requires less memory
- Can do things you can't do in 3D!

3D != better gameplay



Castlevania:
Symphony of the Night

93% Metacritic



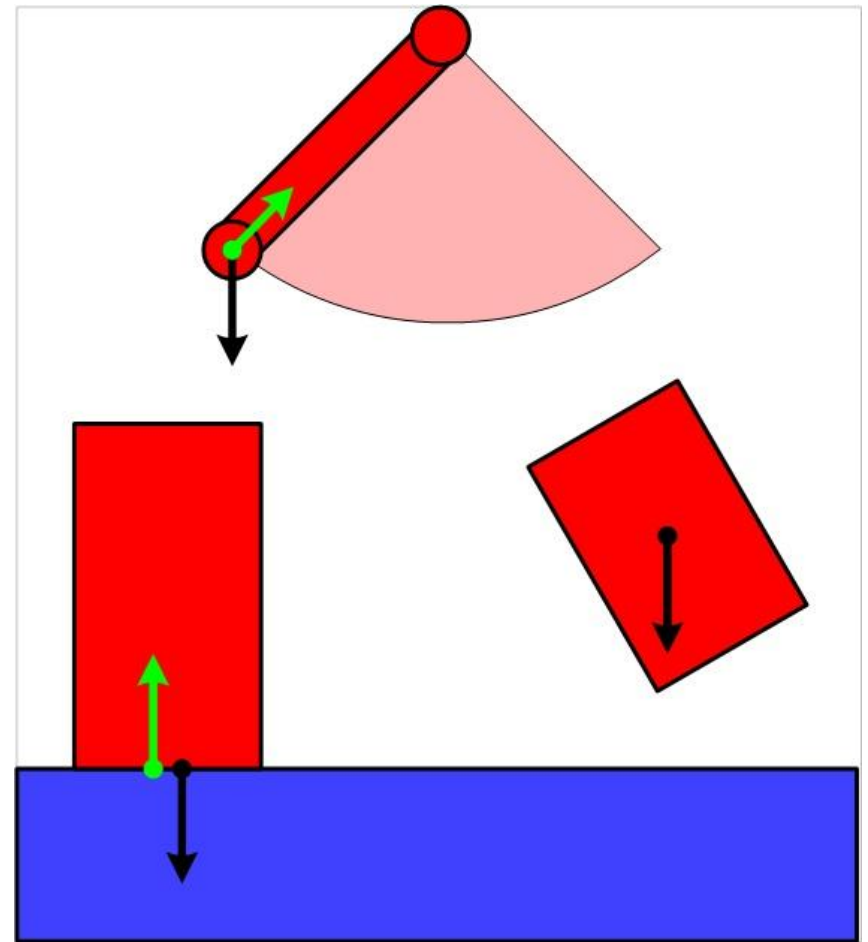
Castlevania:
Judgement

49% Metacritic



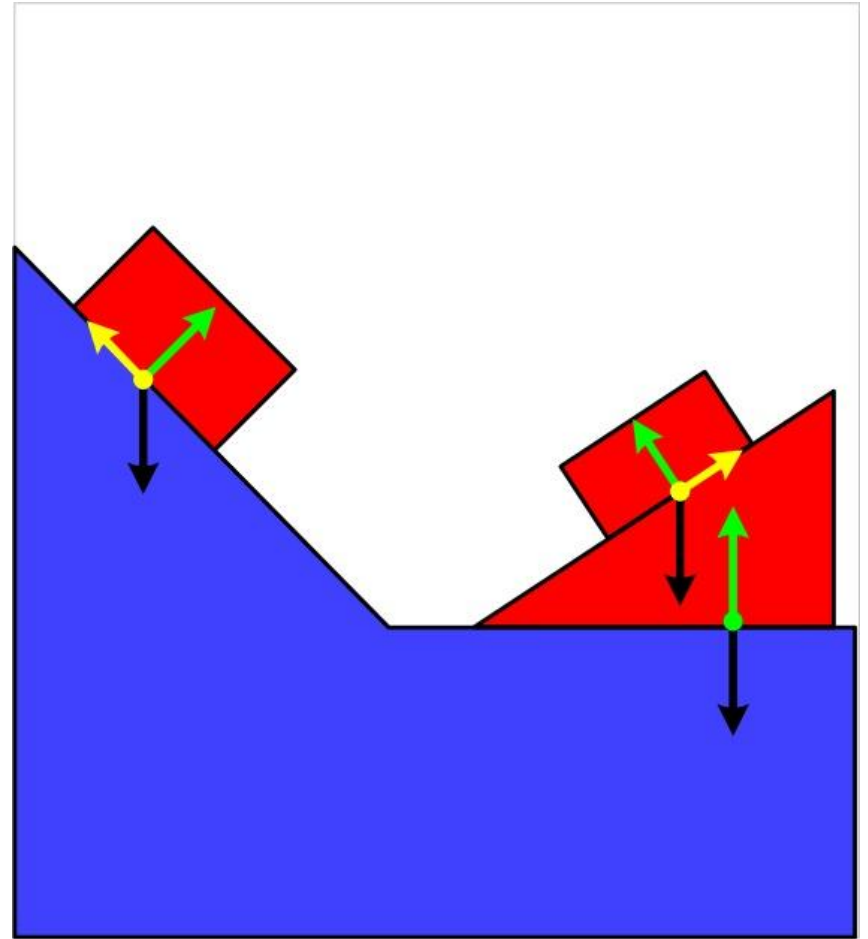
What's the same?

- Position, velocity, mass, momentum, acceleration, force



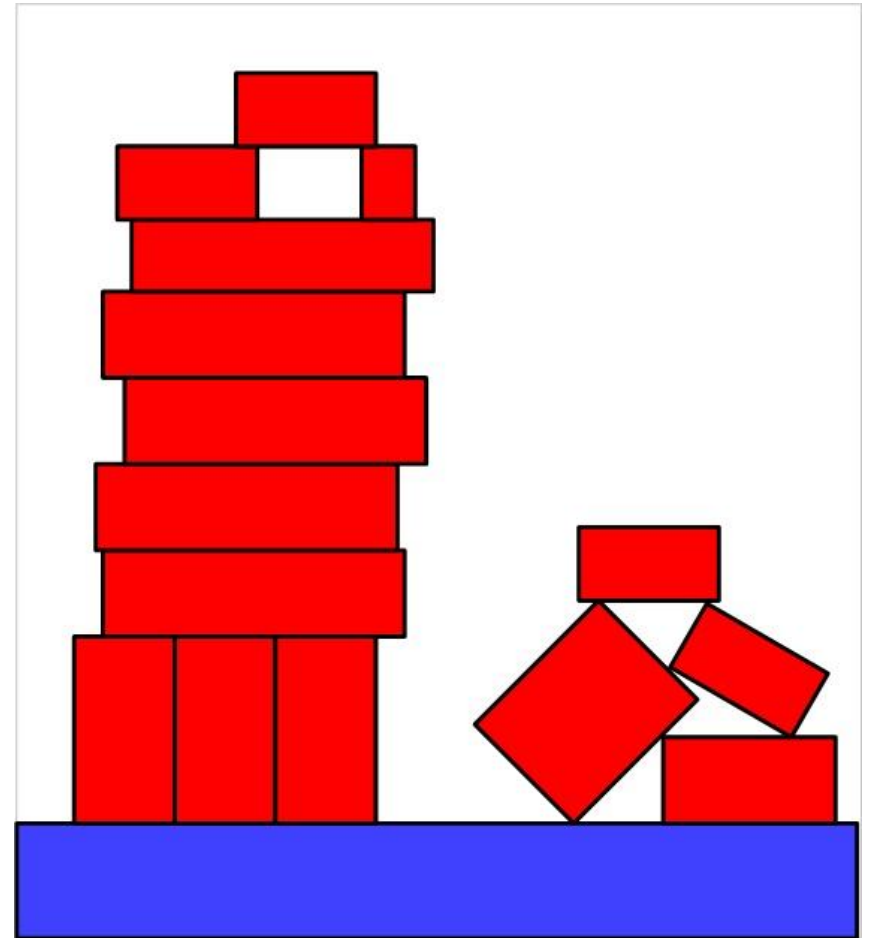
What's the same?

- Position, velocity, mass, momentum, acceleration, force
- Friction



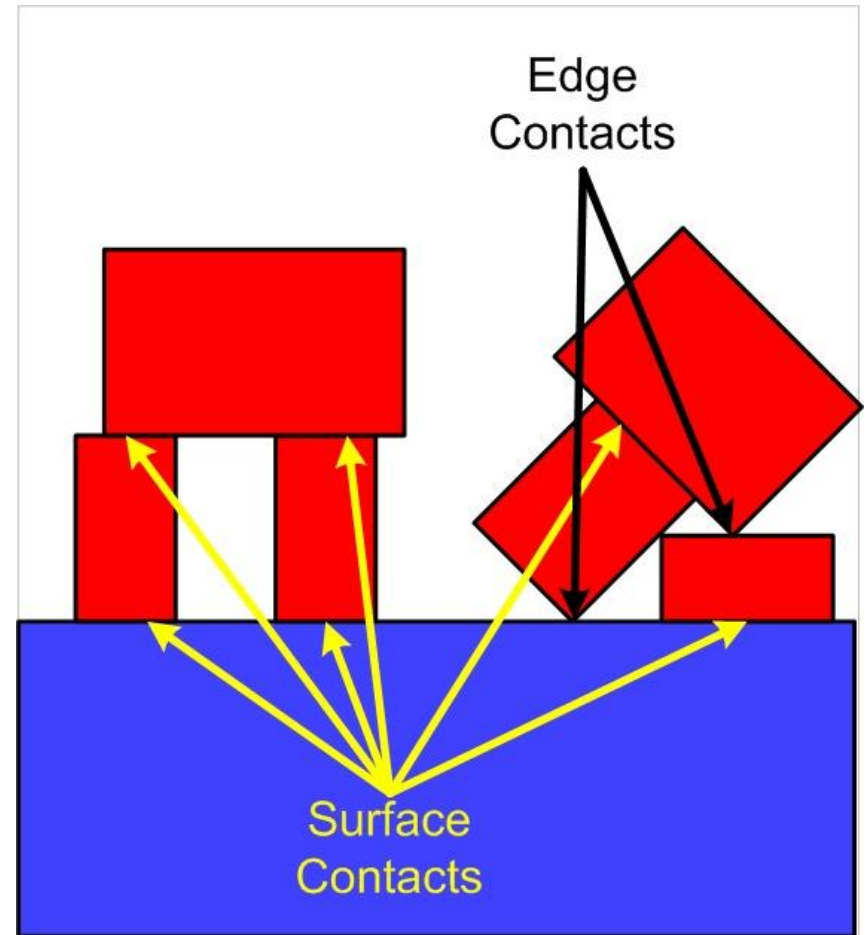
What's the same?

- Position, velocity, mass, momentum, acceleration, force
- Friction
- Stacking



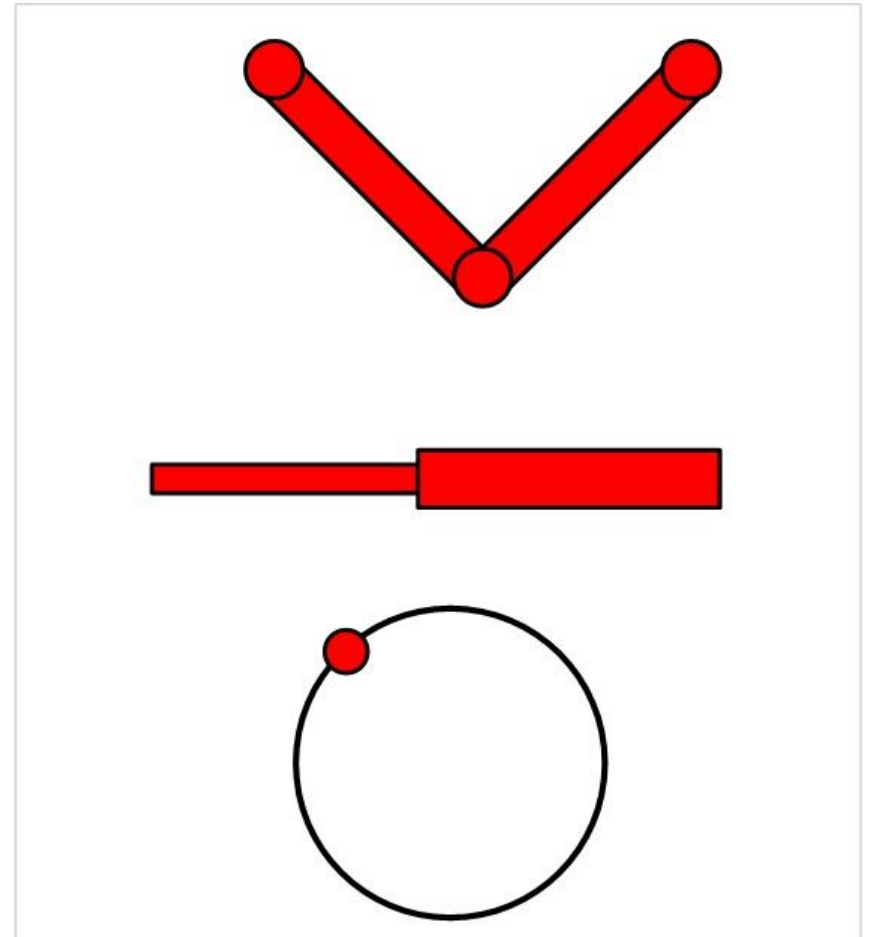
What's the same?

- Position, velocity, mass, momentum, acceleration, force
- Friction
- Stacking
- Contact



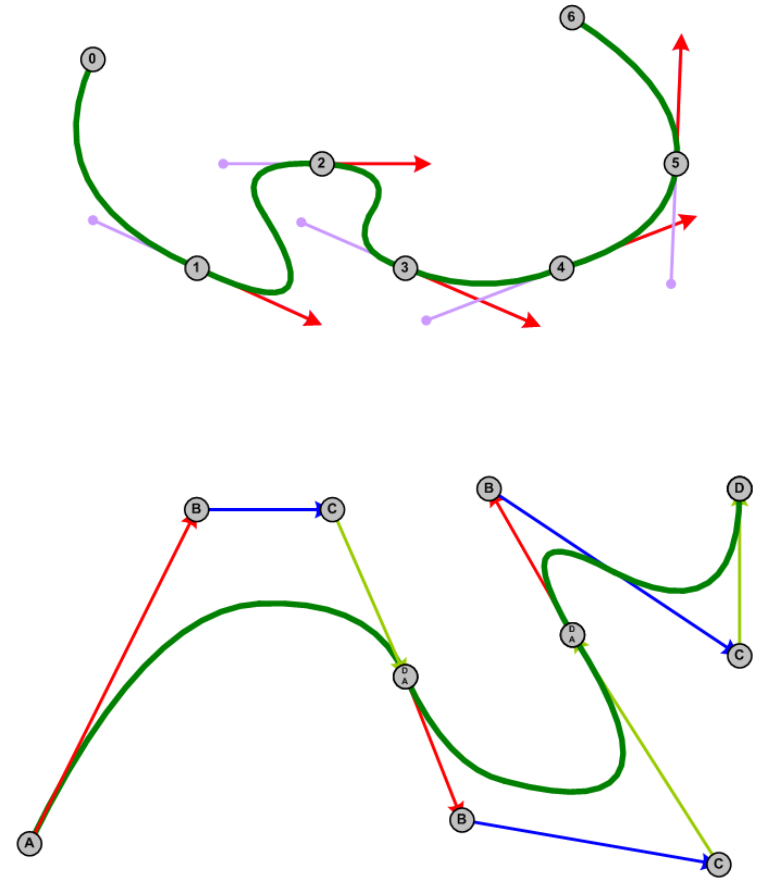
What's the same?

- Position, velocity, mass, momentum, acceleration, force
- Friction
- Stacking
- Contact
- Joints & constraints



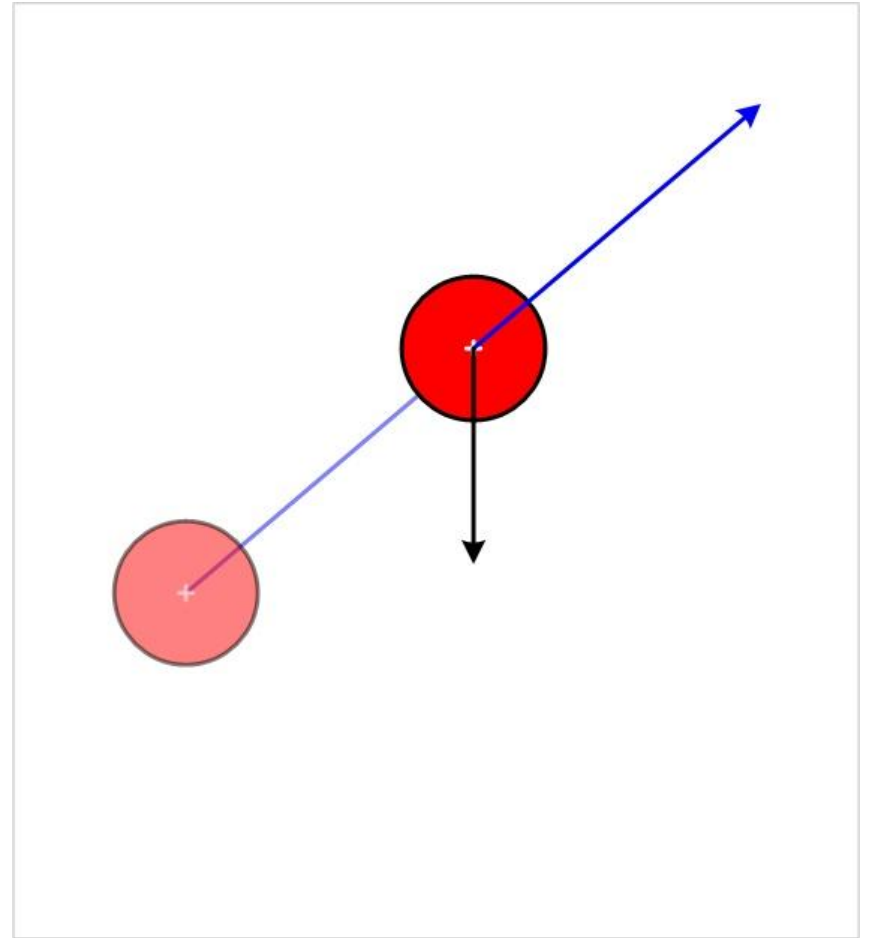
What's the same?

- Curves and splines



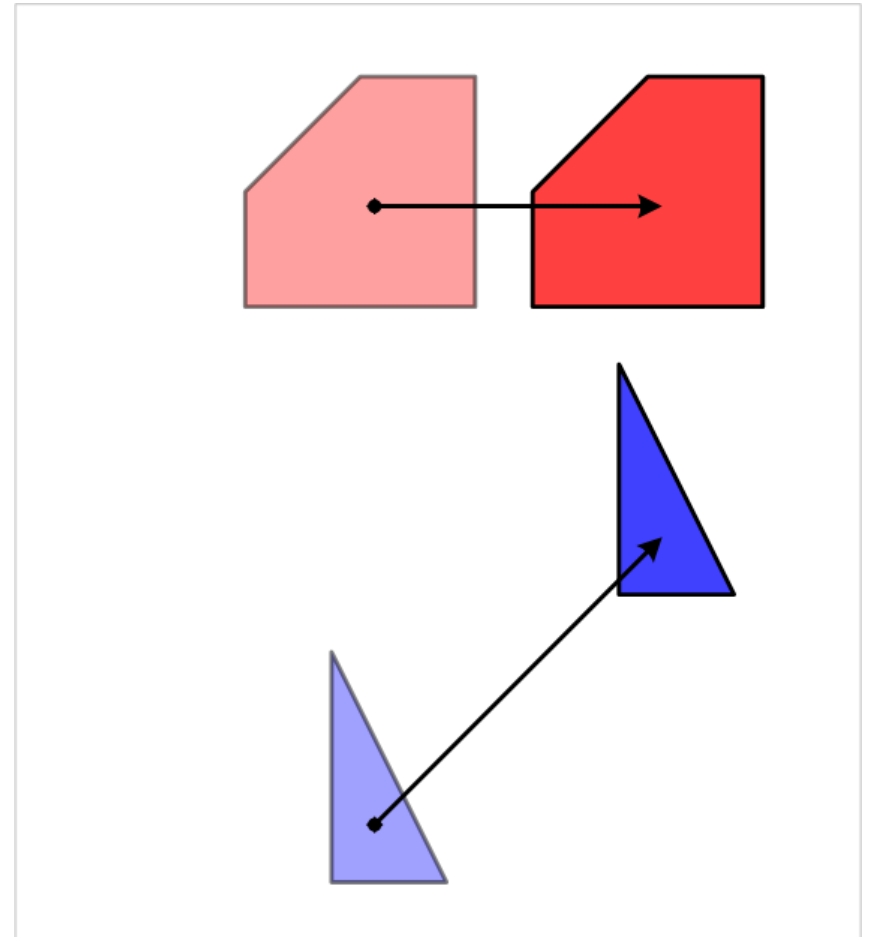
What's the same?

- Curves and splines
- Numerical integration



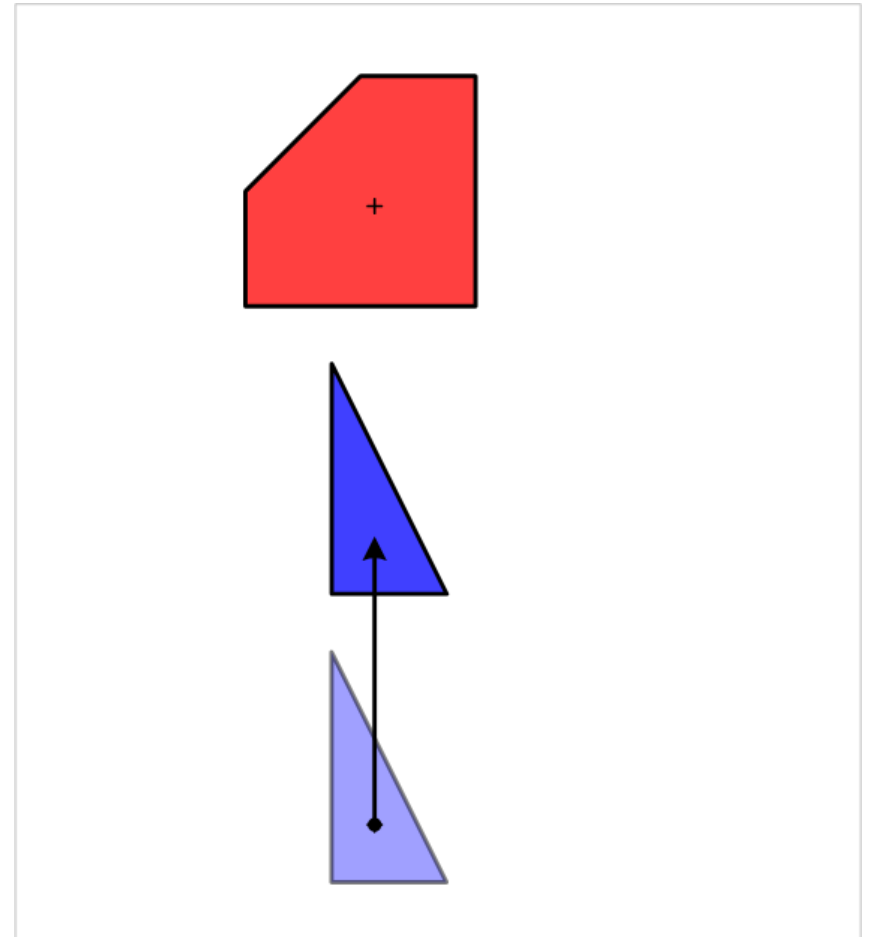
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference



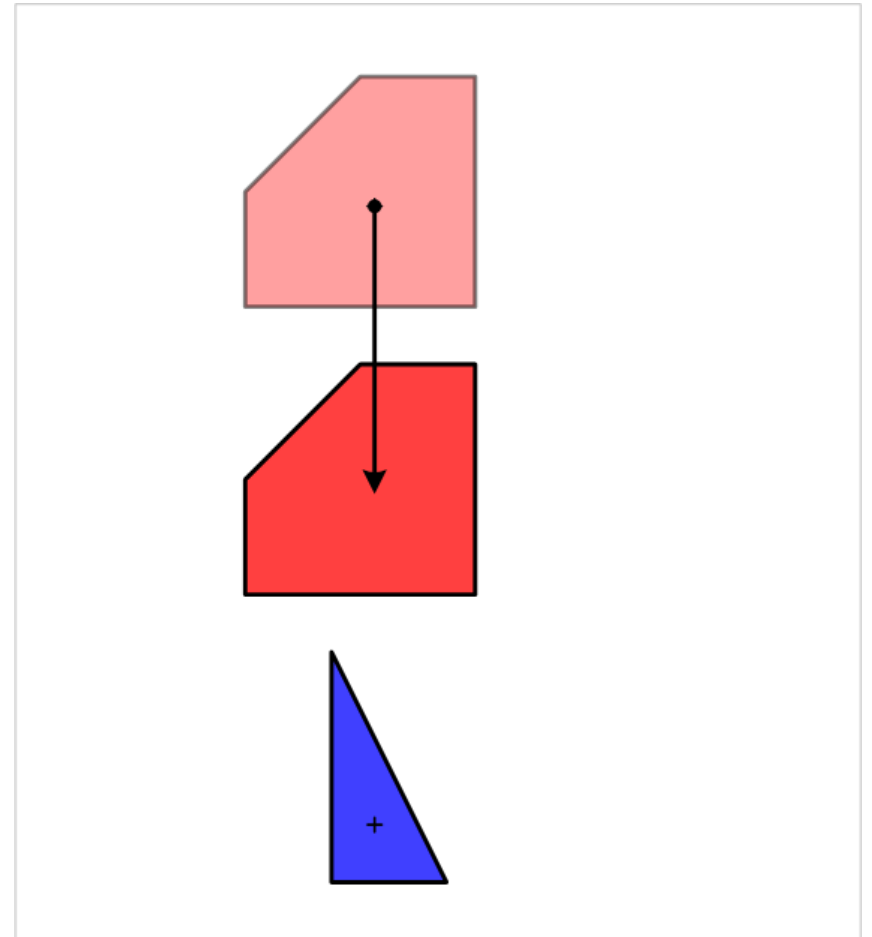
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference



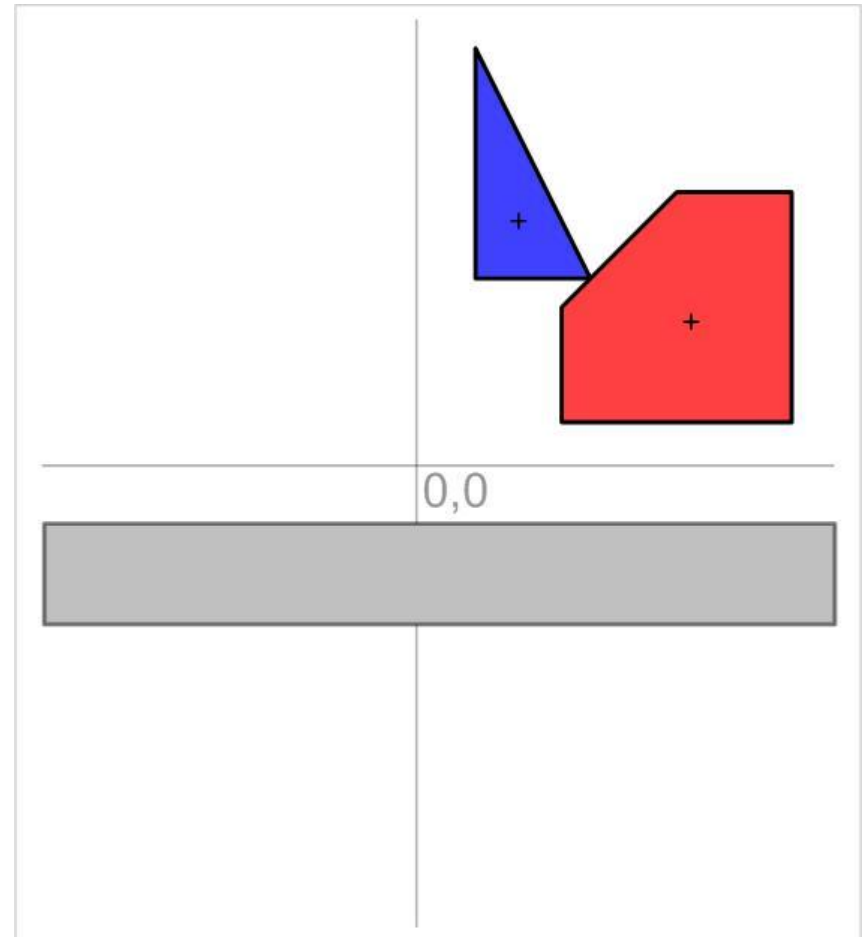
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference



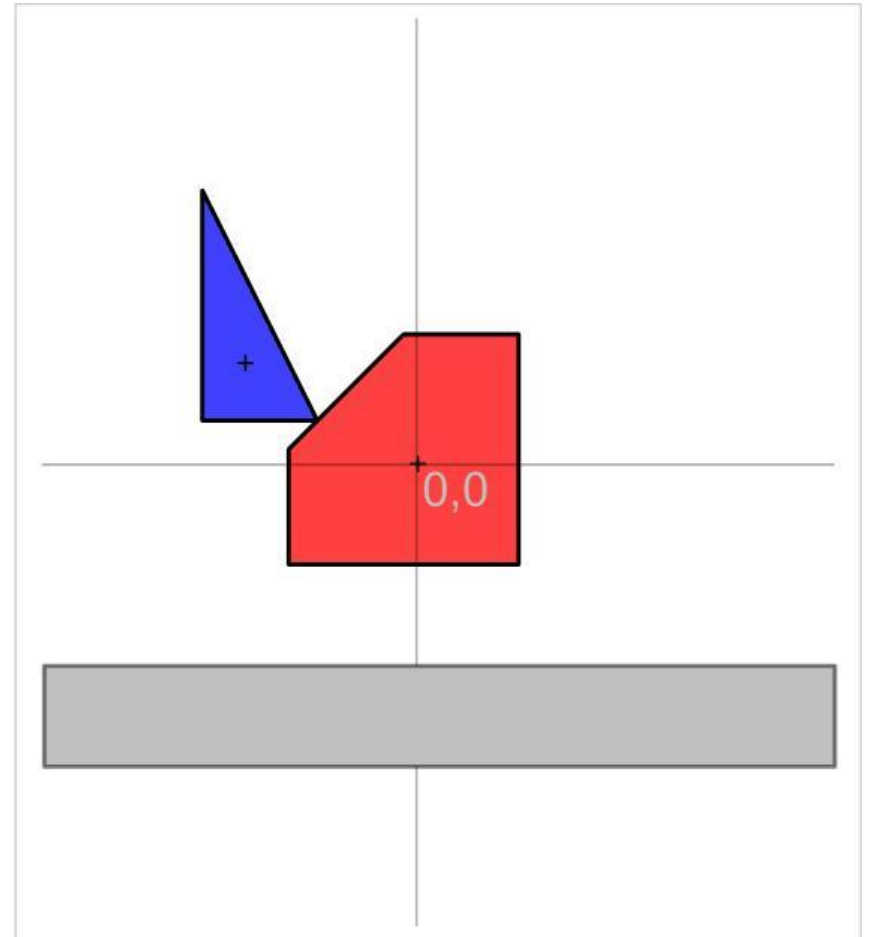
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



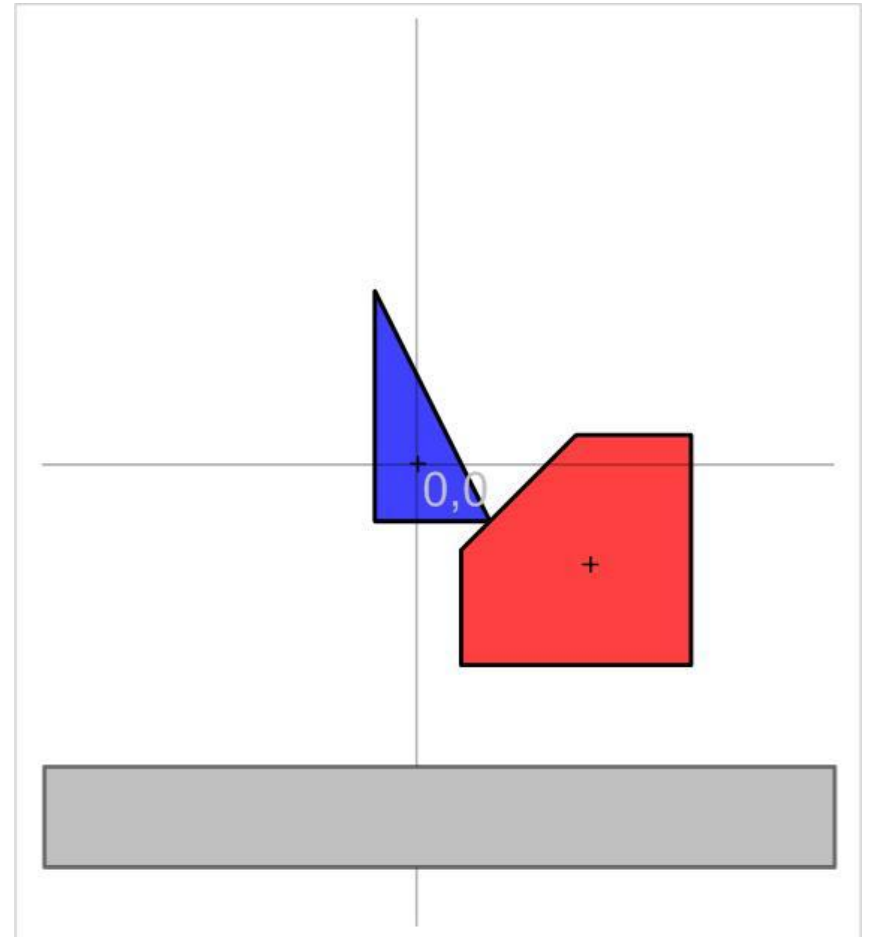
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



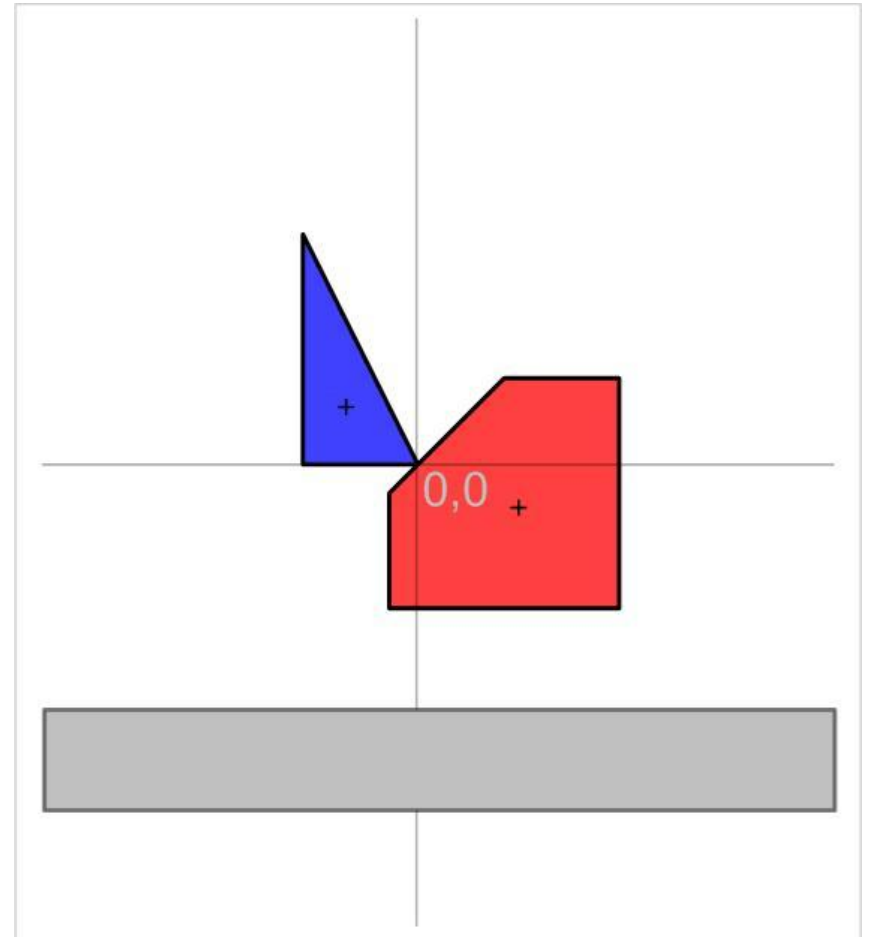
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



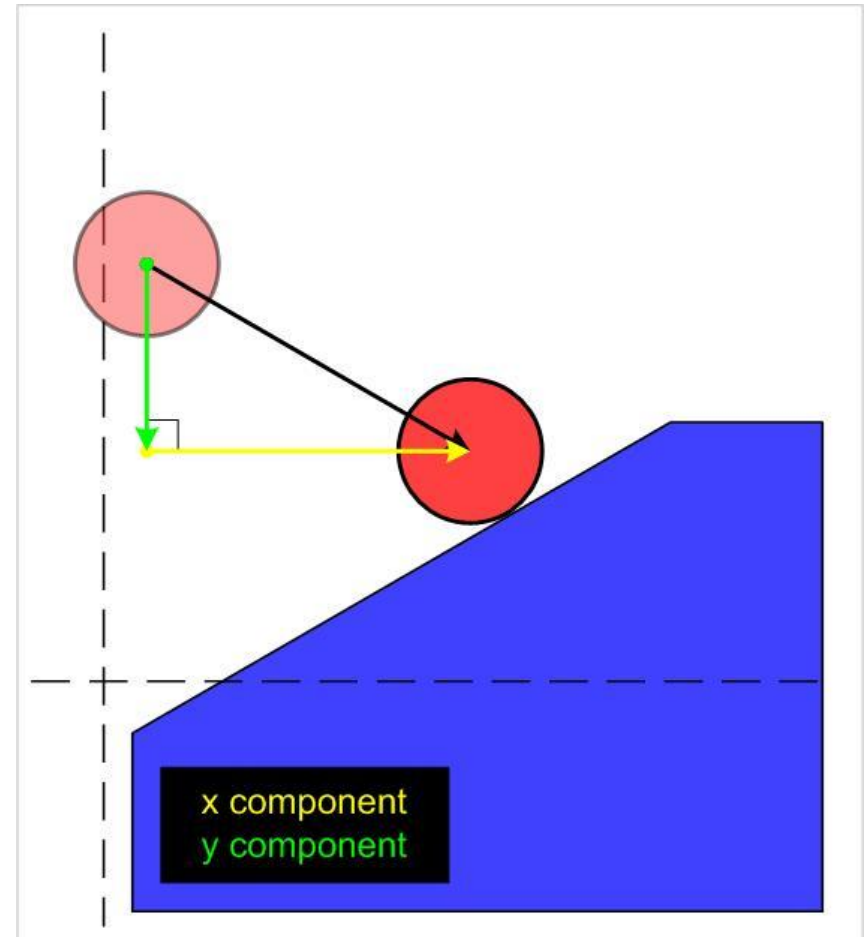
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



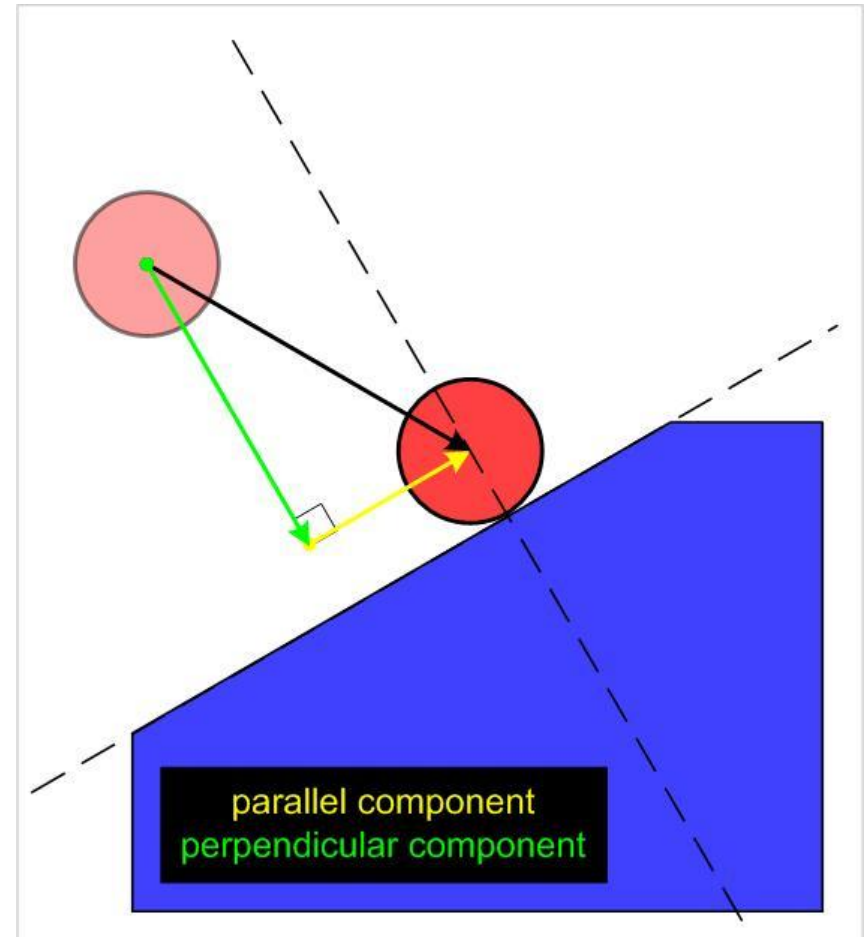
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



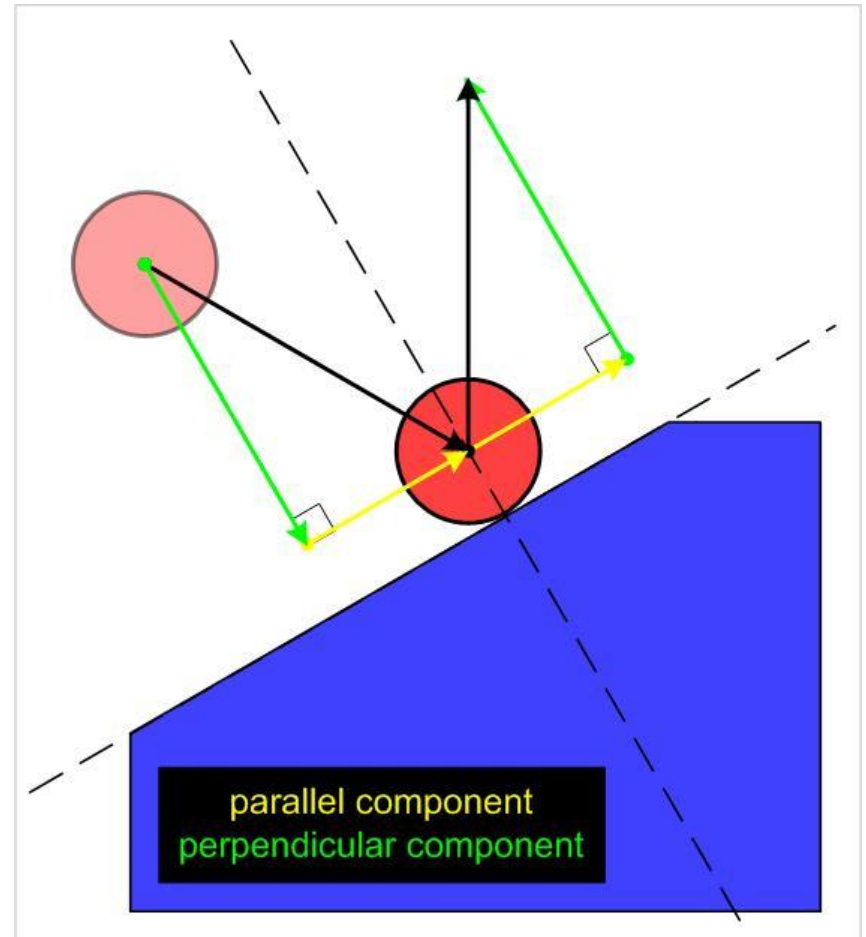
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



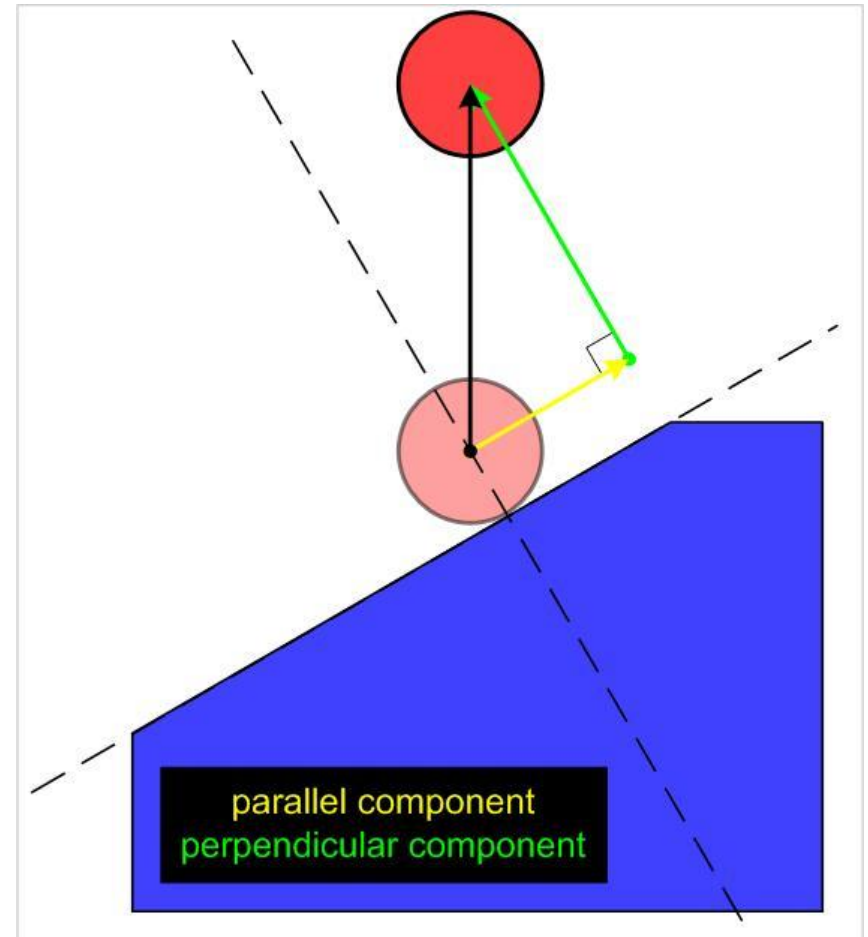
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



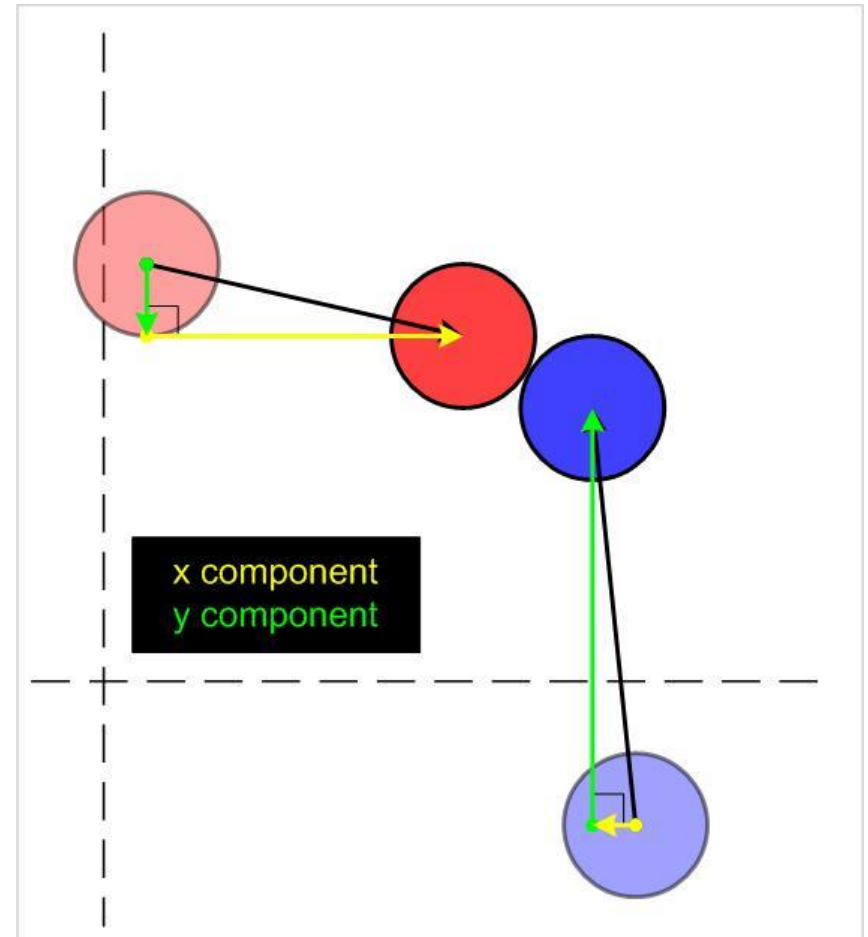
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



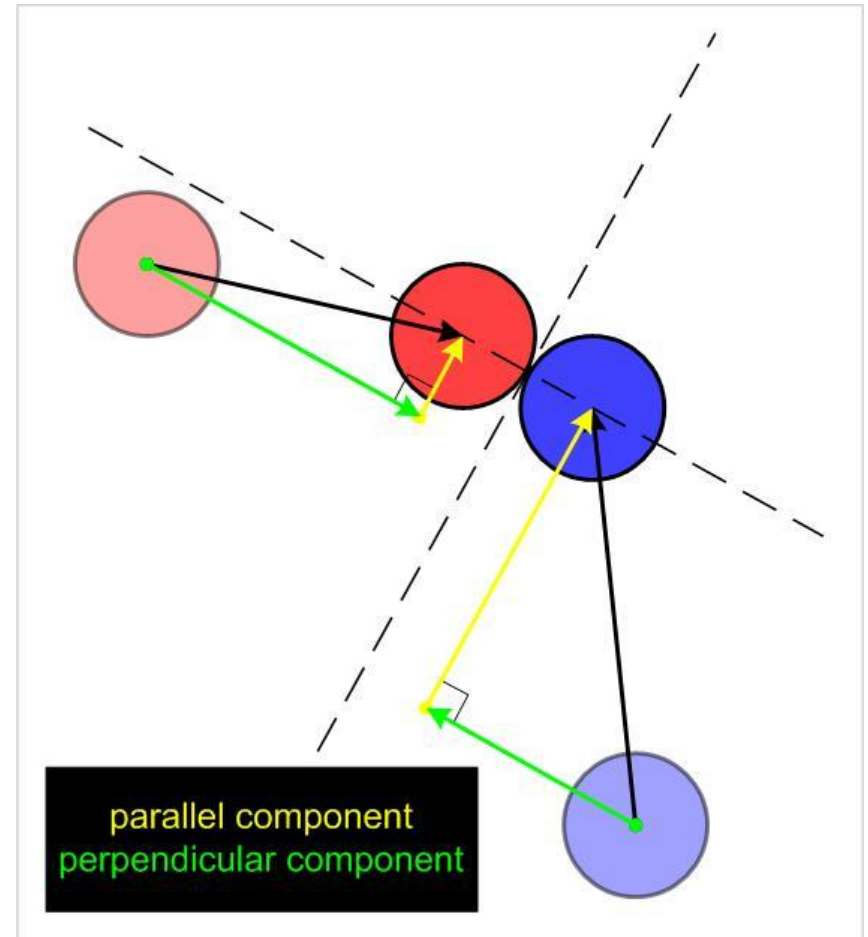
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



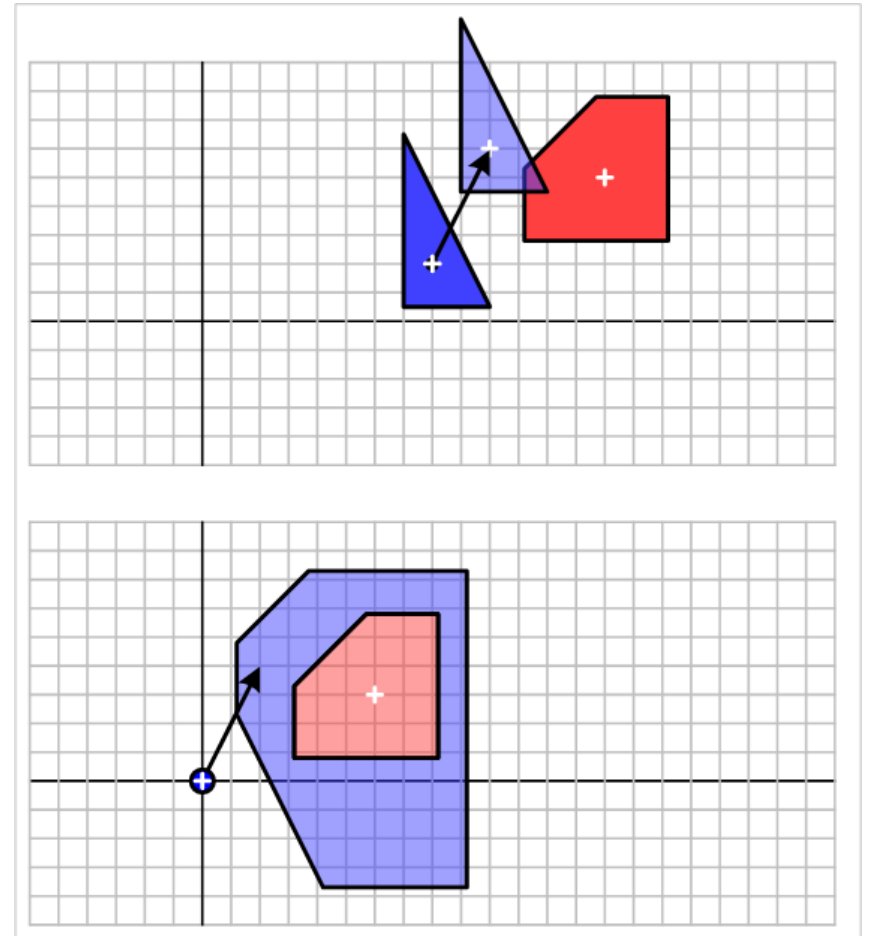
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems



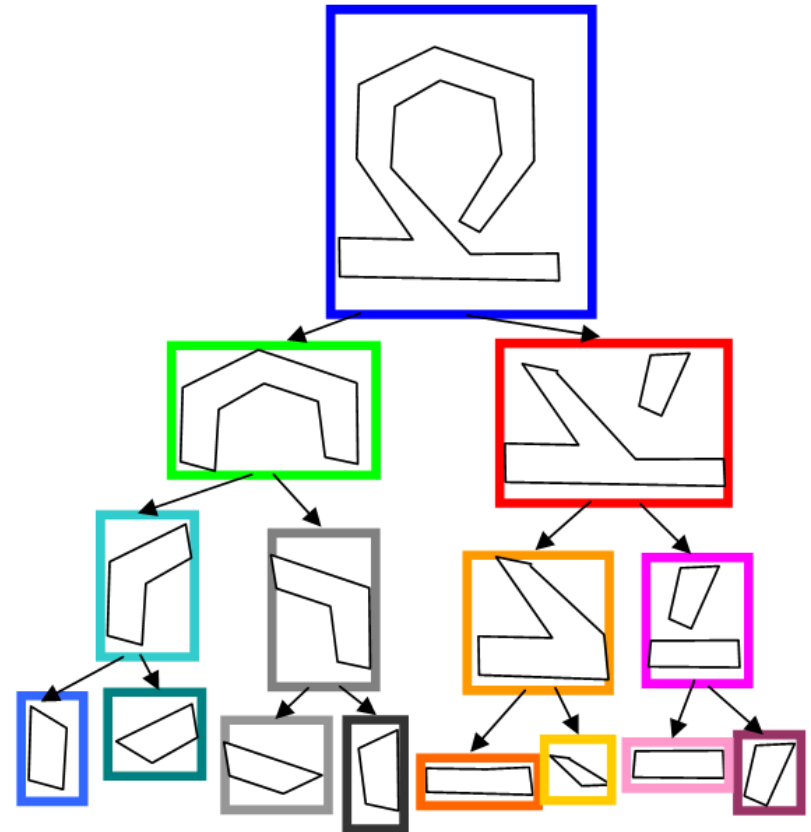
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems
- Configuration Space



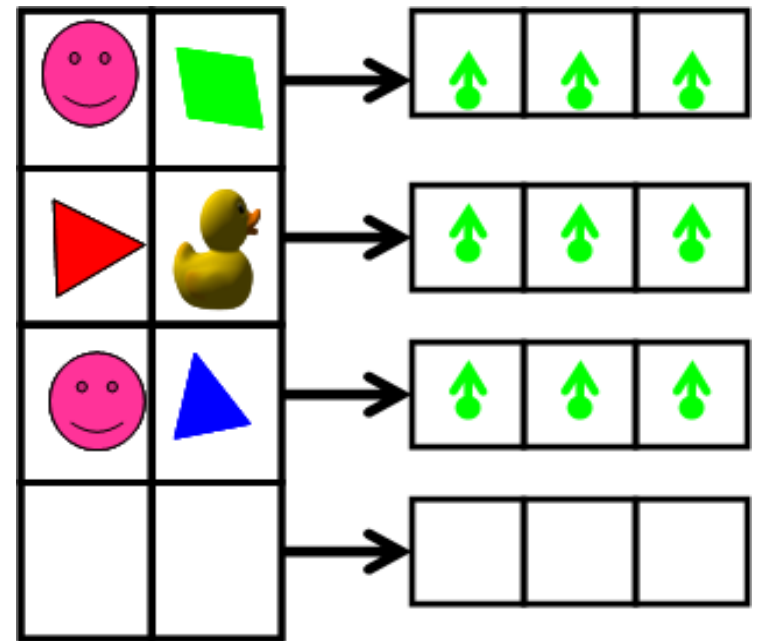
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems
- Configuration Space
- Object hierarchies



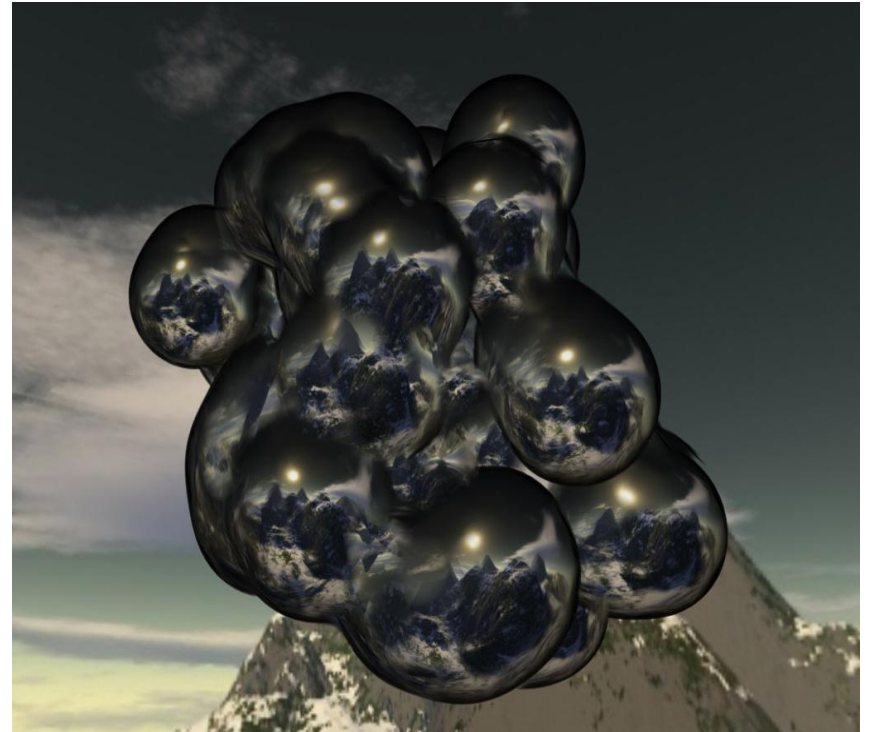
What's the same?

- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems
- Configuration Space
- Object hierarchies
- Parallelisms



What's the same?

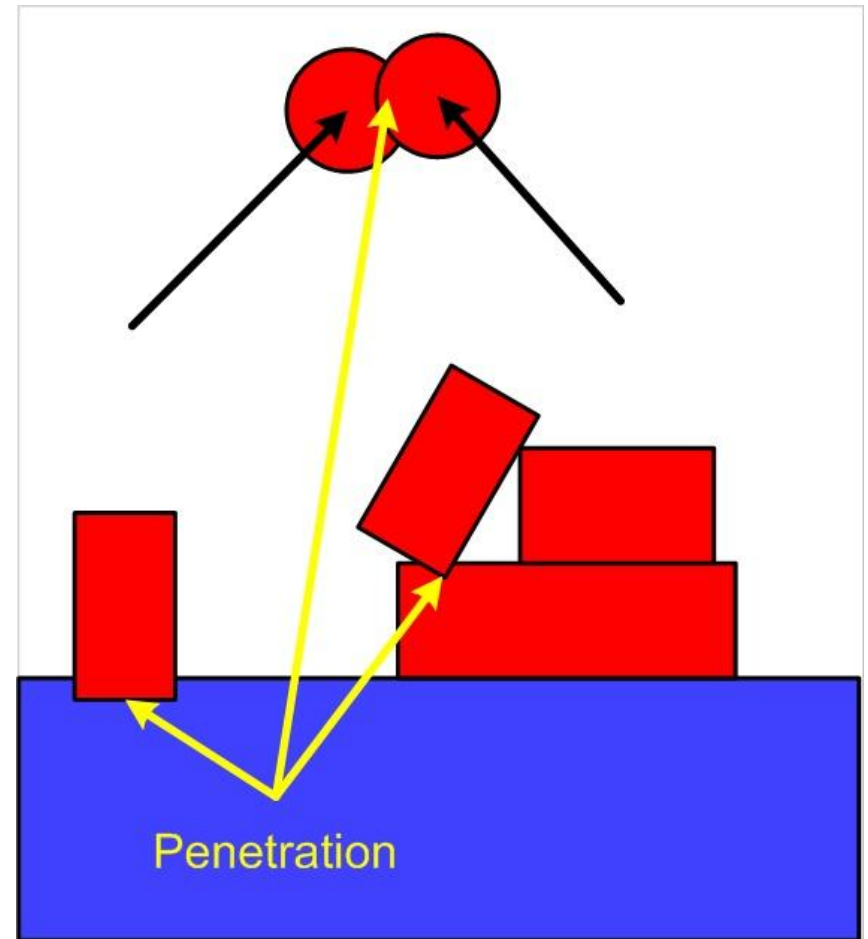
- Curves and splines
- Numerical integration
- Frames of Reference
- Coordinate systems
- Configuration Space
- Object hierarchies
- Parallelisms
- Fluids, cloth, etc.



What's the same?

Similar problems:

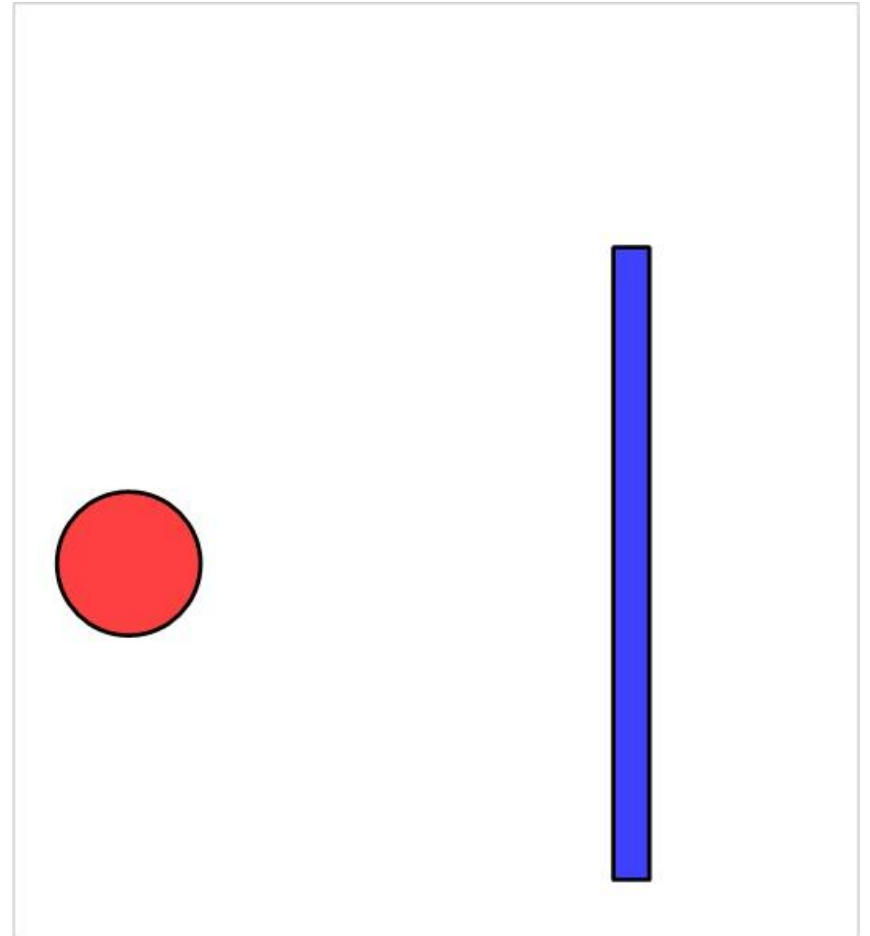
- Penetration



What's the same?

Similar problems:

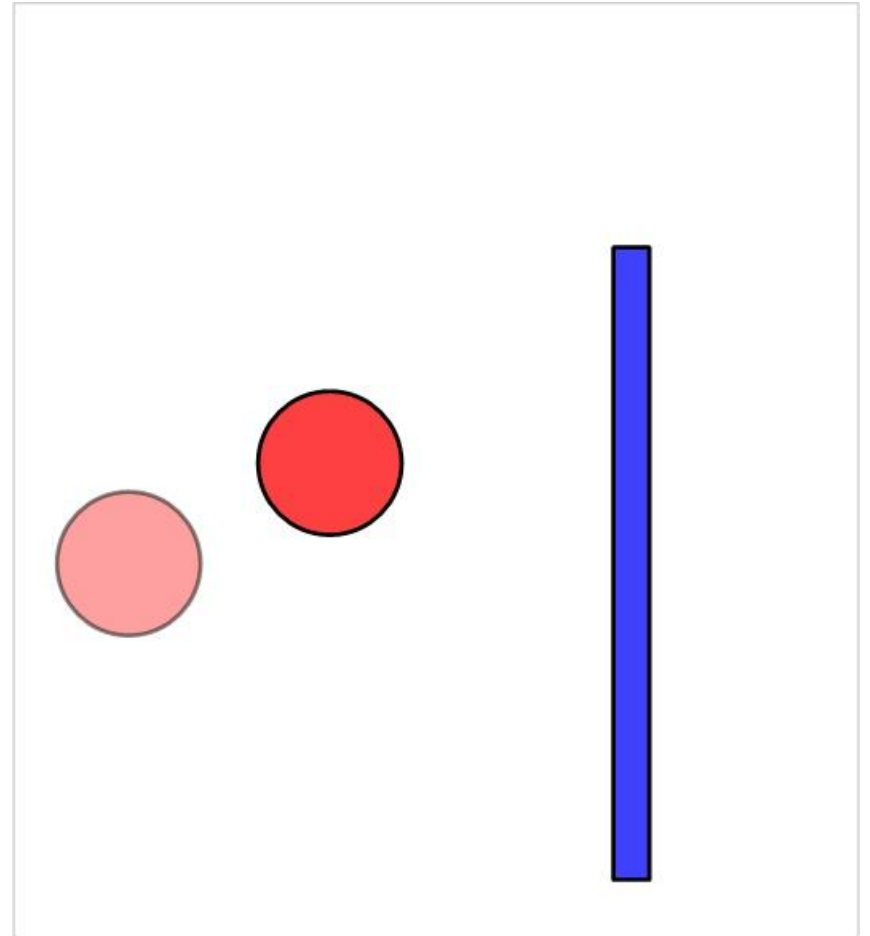
- Penetration
- Tunneling



What's the same?

Similar problems:

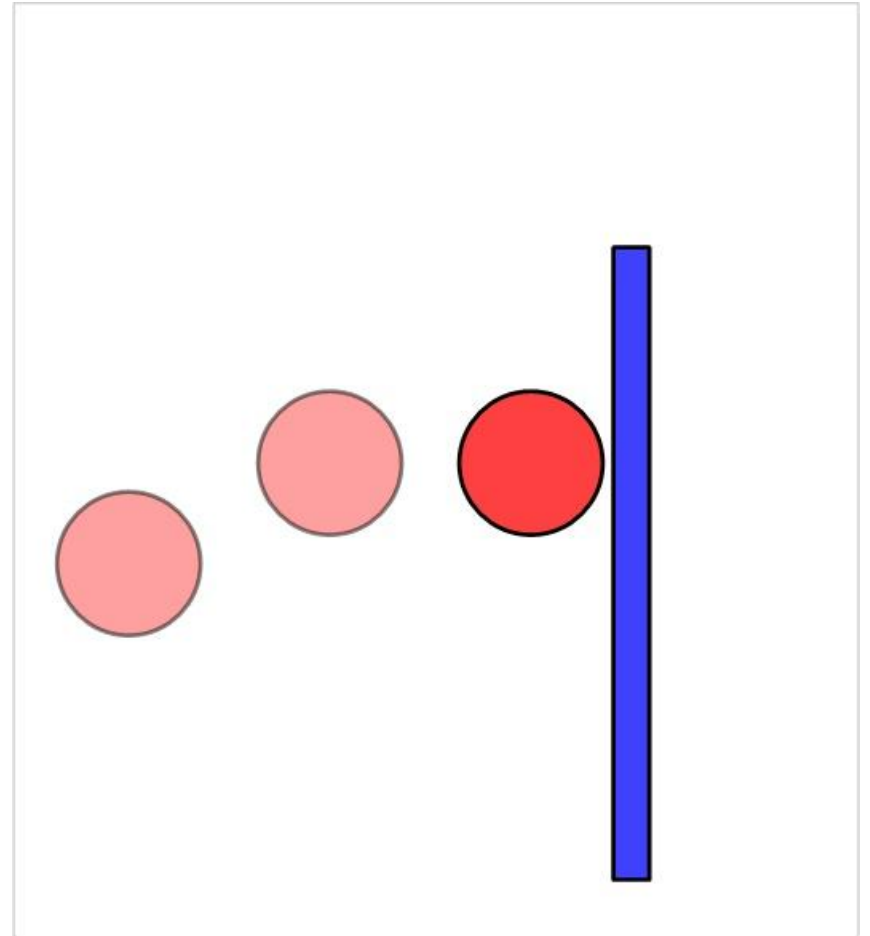
- Penetration
- Tunneling



What's the same?

Similar problems:

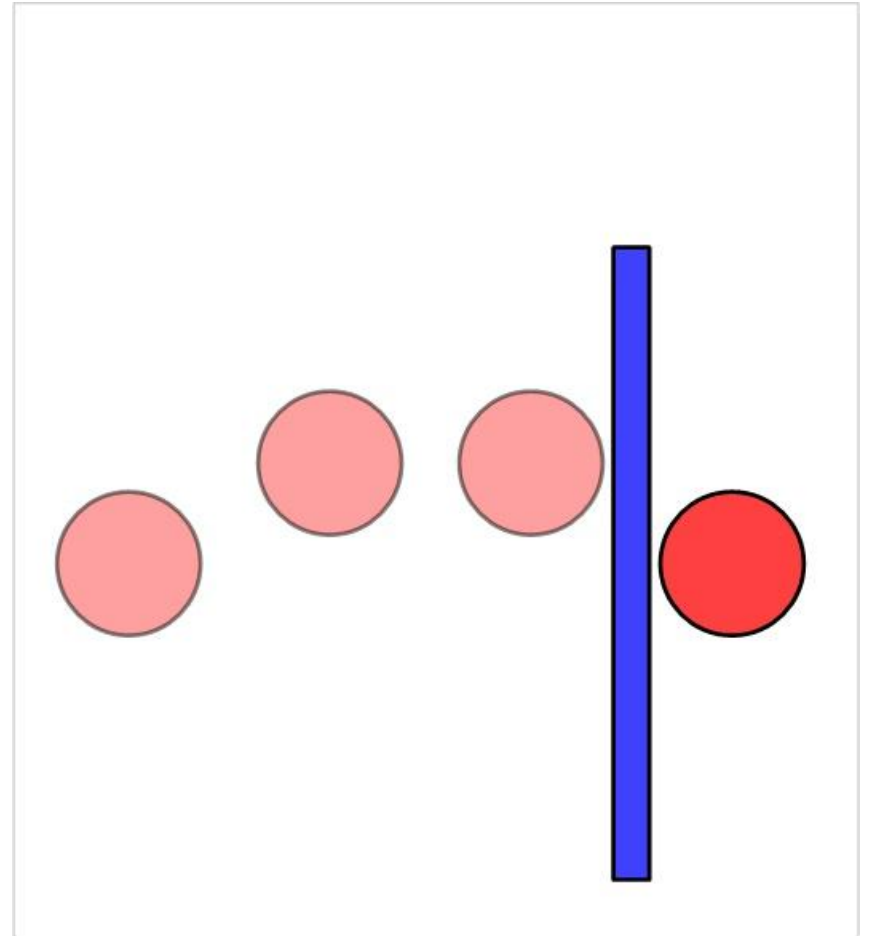
- Penetration
- Tunneling



What's the same?

Similar problems:

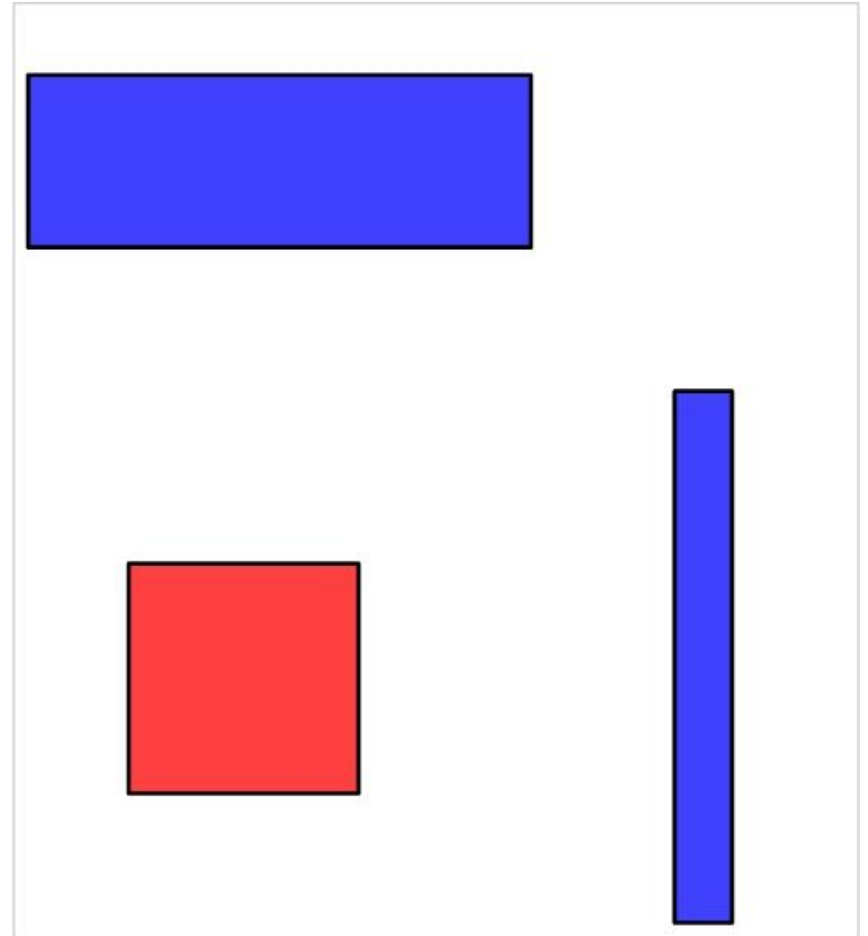
- Penetration
- Tunneling



What's the same?

Similar problems:

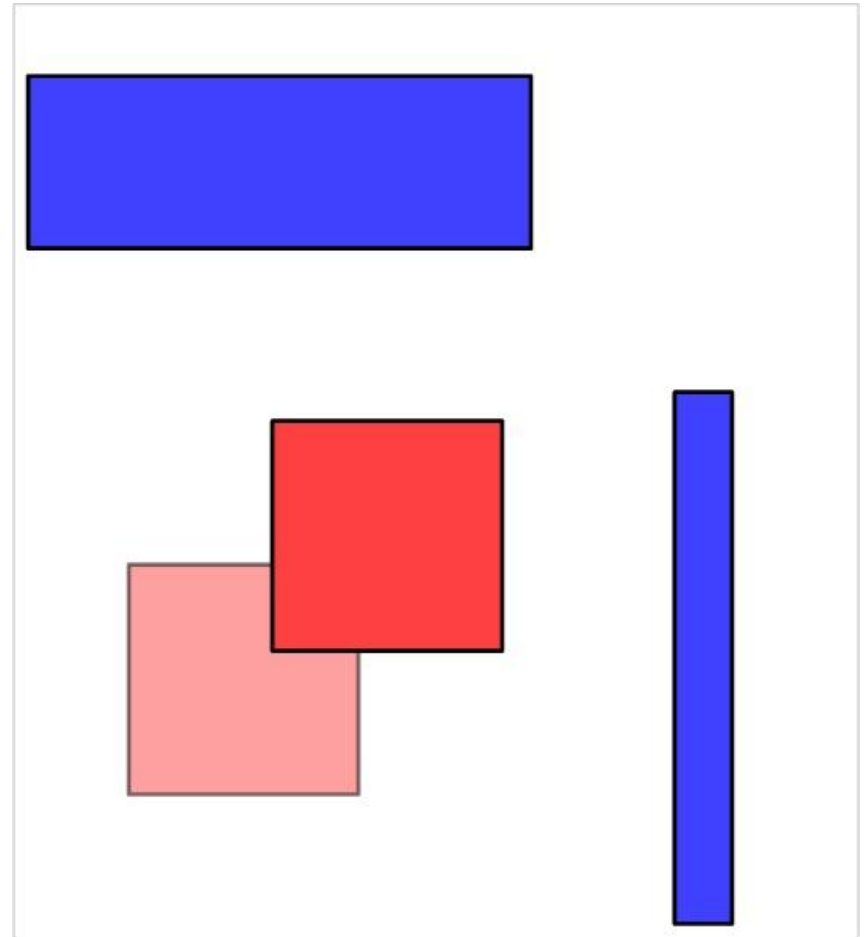
- Penetration
- Tunneling



What's the same?

Similar problems:

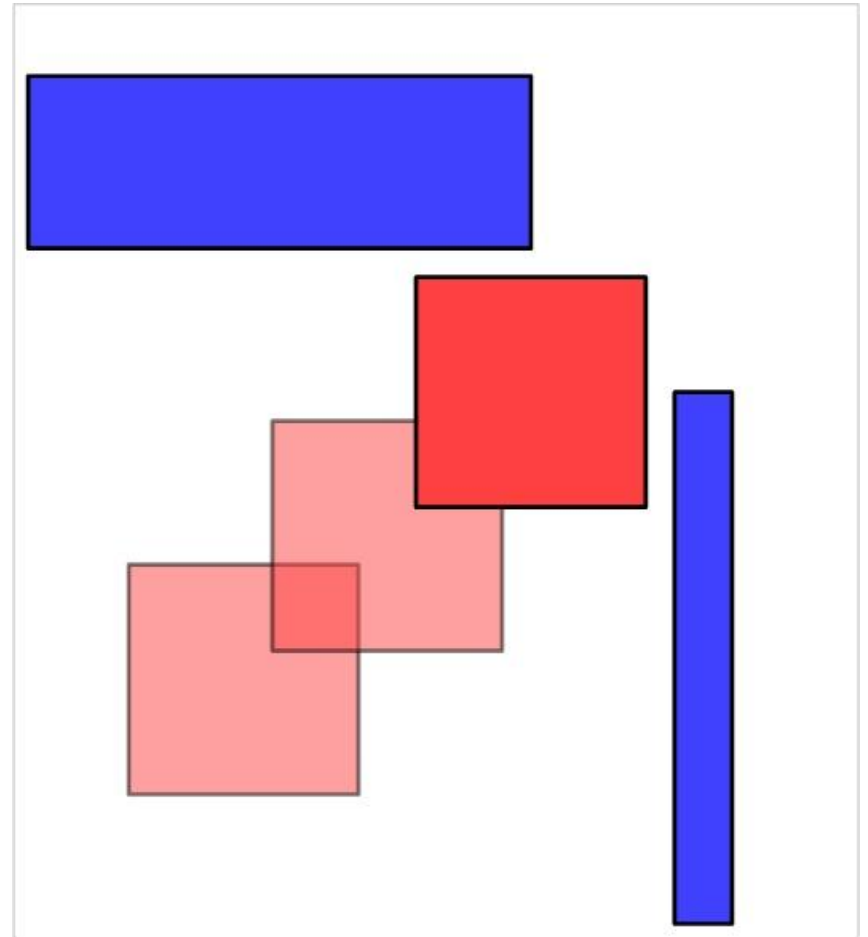
- Penetration
- Tunneling



What's the same?

Similar problems:

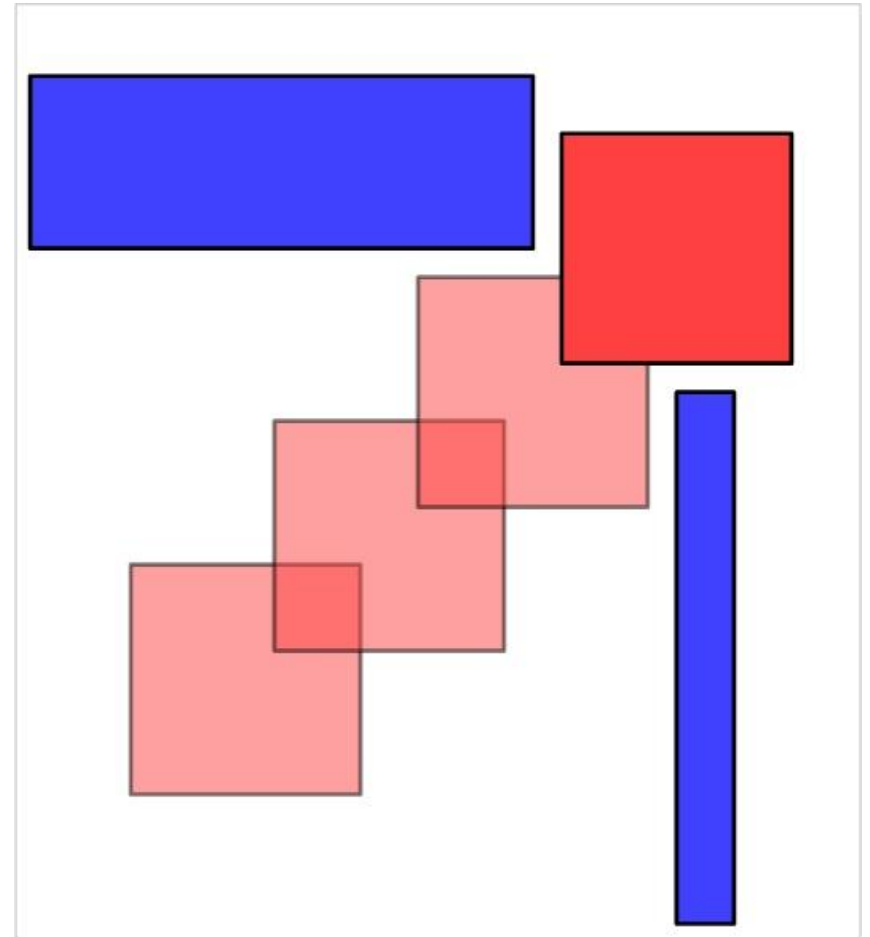
- Penetration
- Tunneling



What's the same?

Similar problems:

- Penetration
- Tunneling



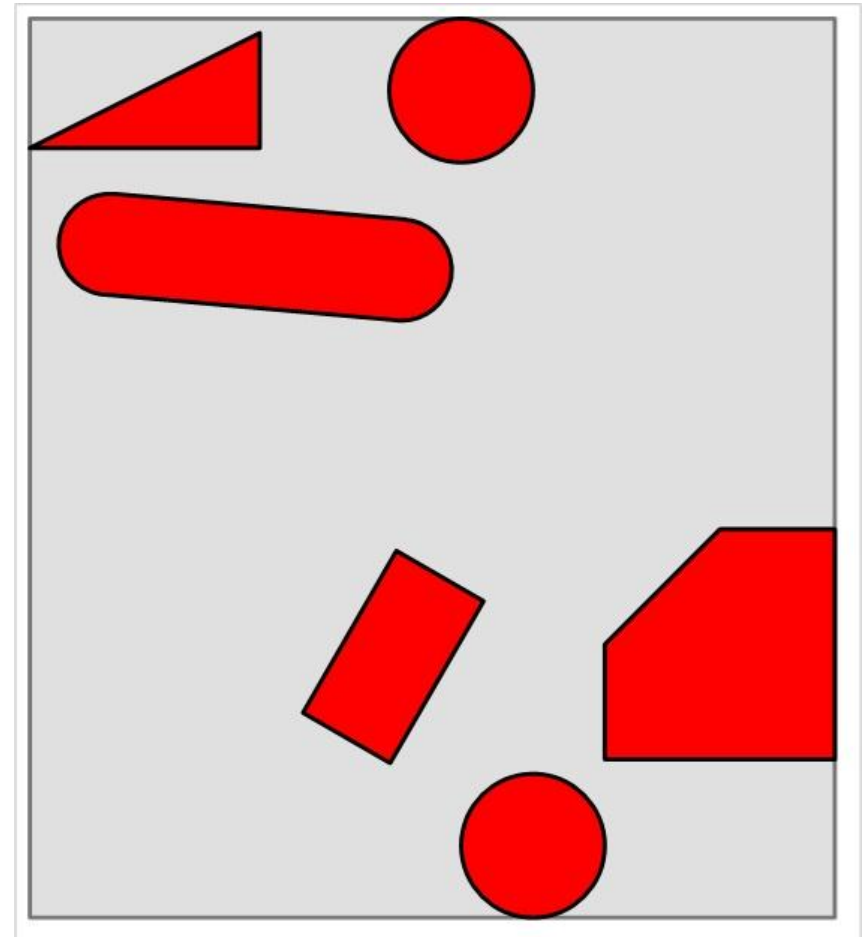
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands
- 1 x (1000x1000)
= 1 million



What's the same?

Similar problems:

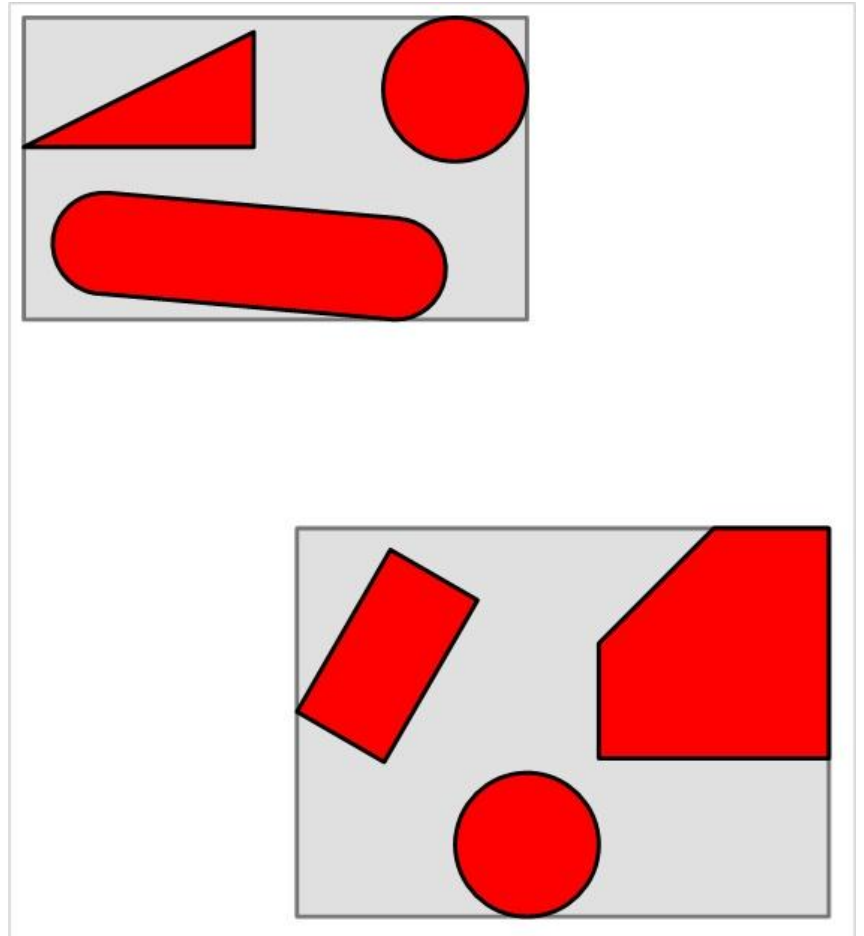
- Penetration
- Tunneling

Similar techniques:

- Simulation islands

10 x (100x100)

= 100,000



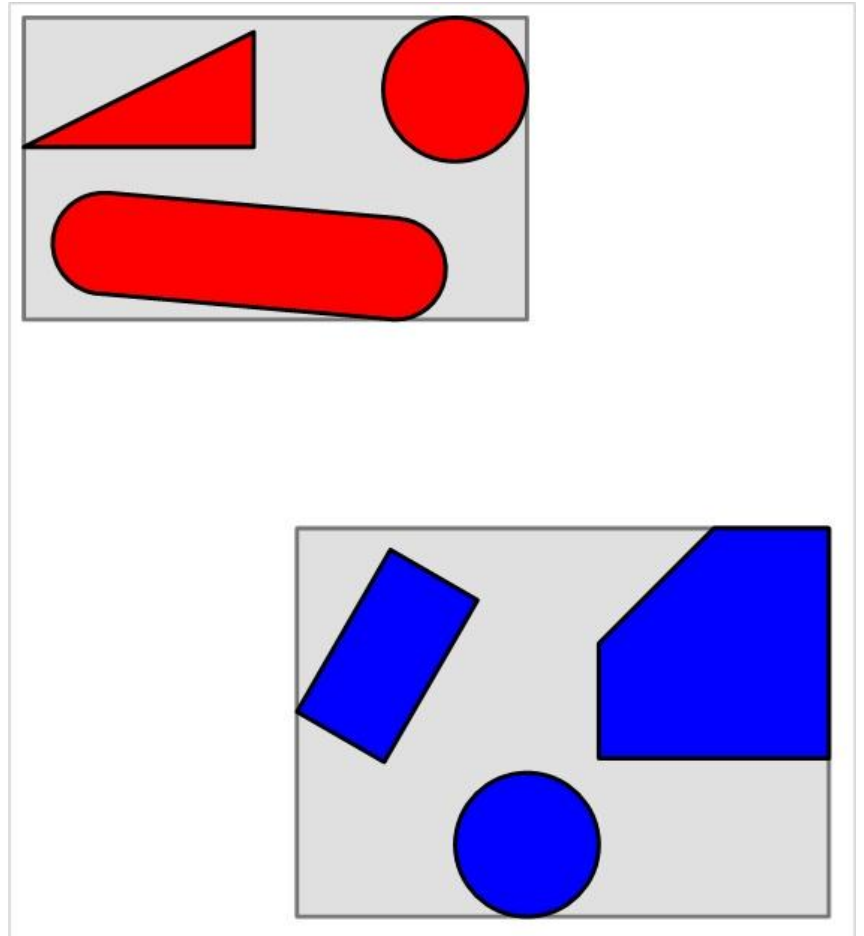
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



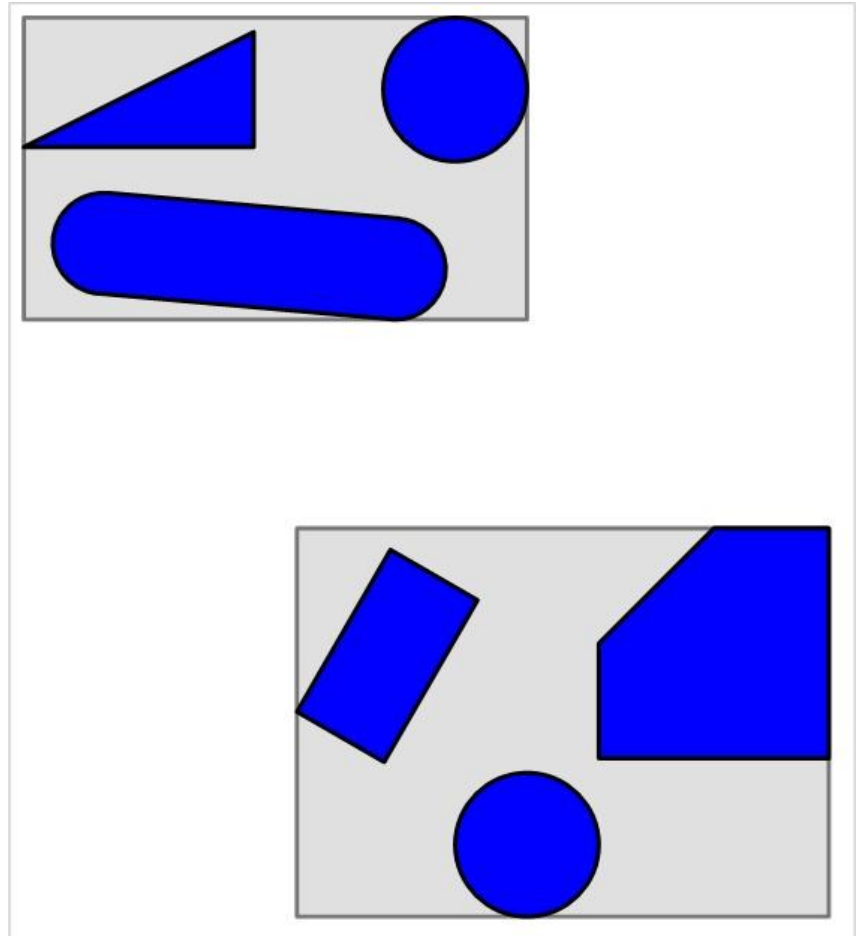
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



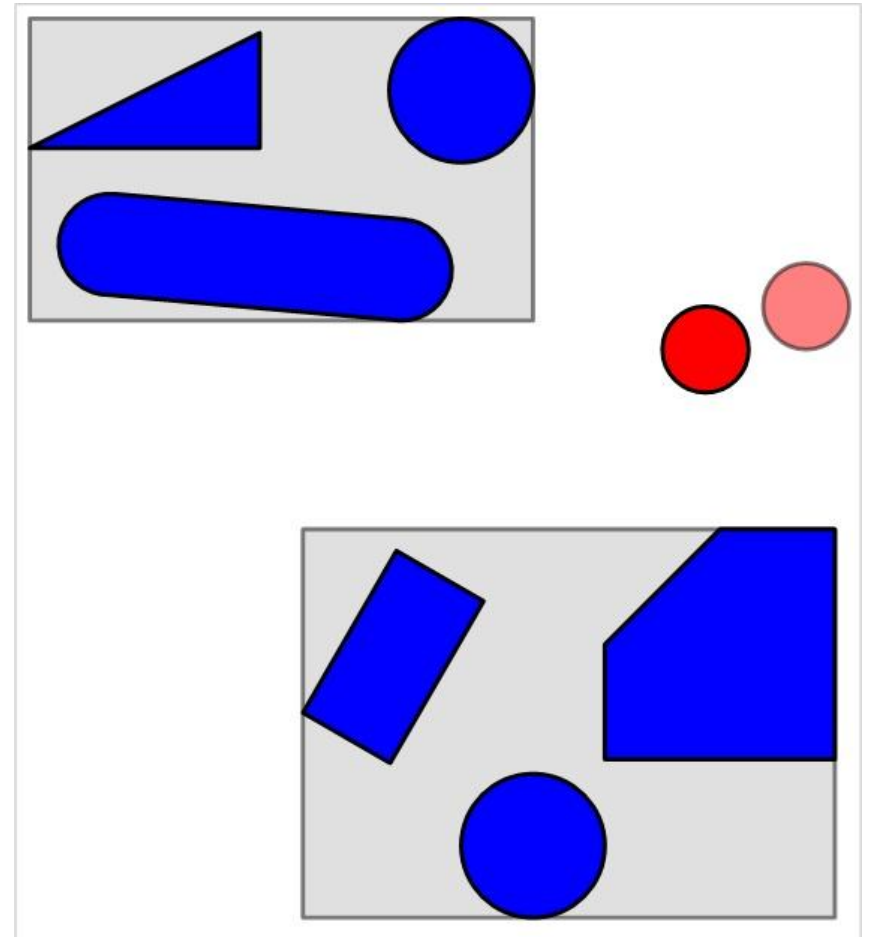
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



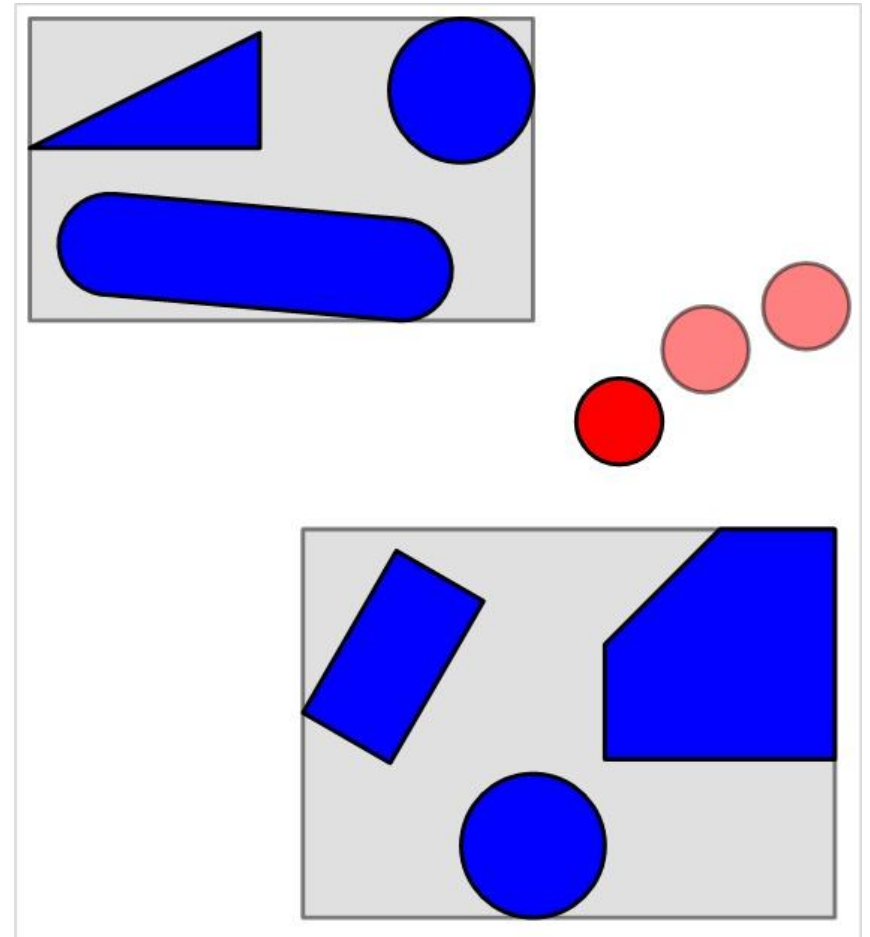
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



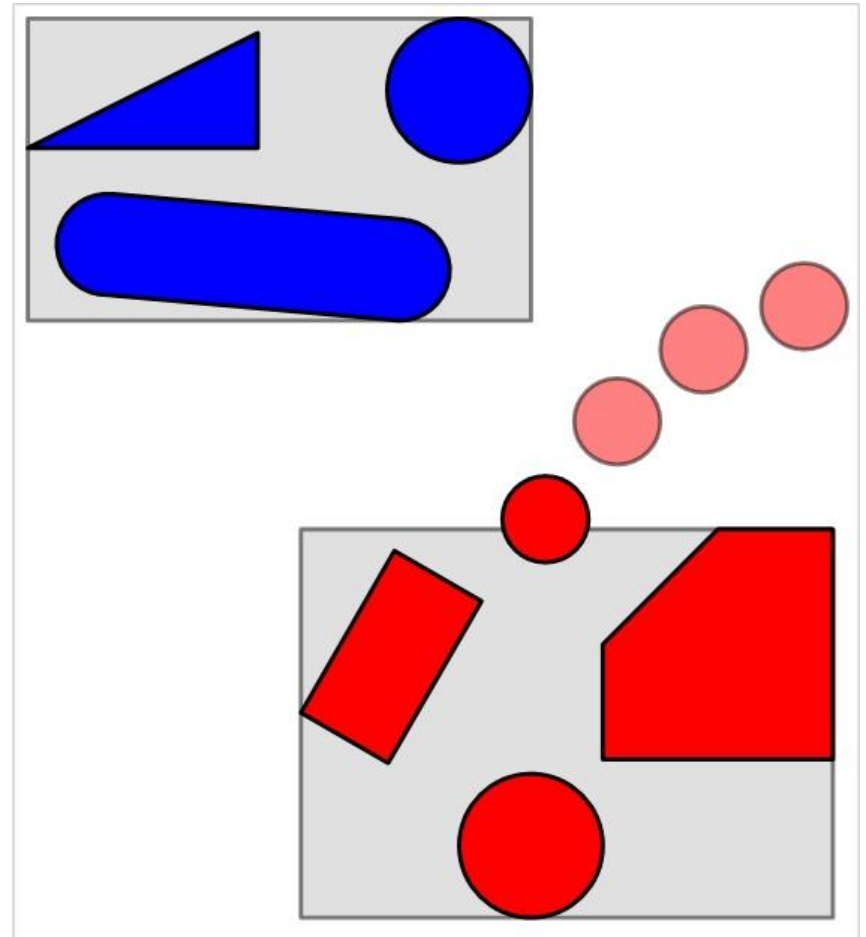
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



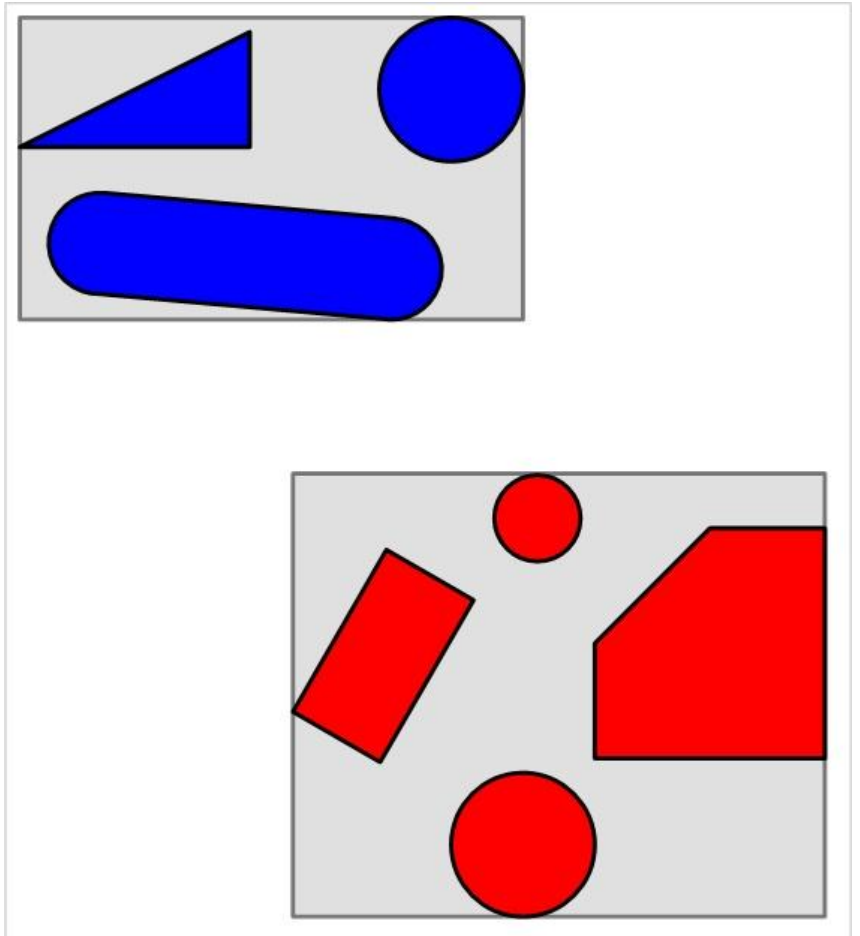
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



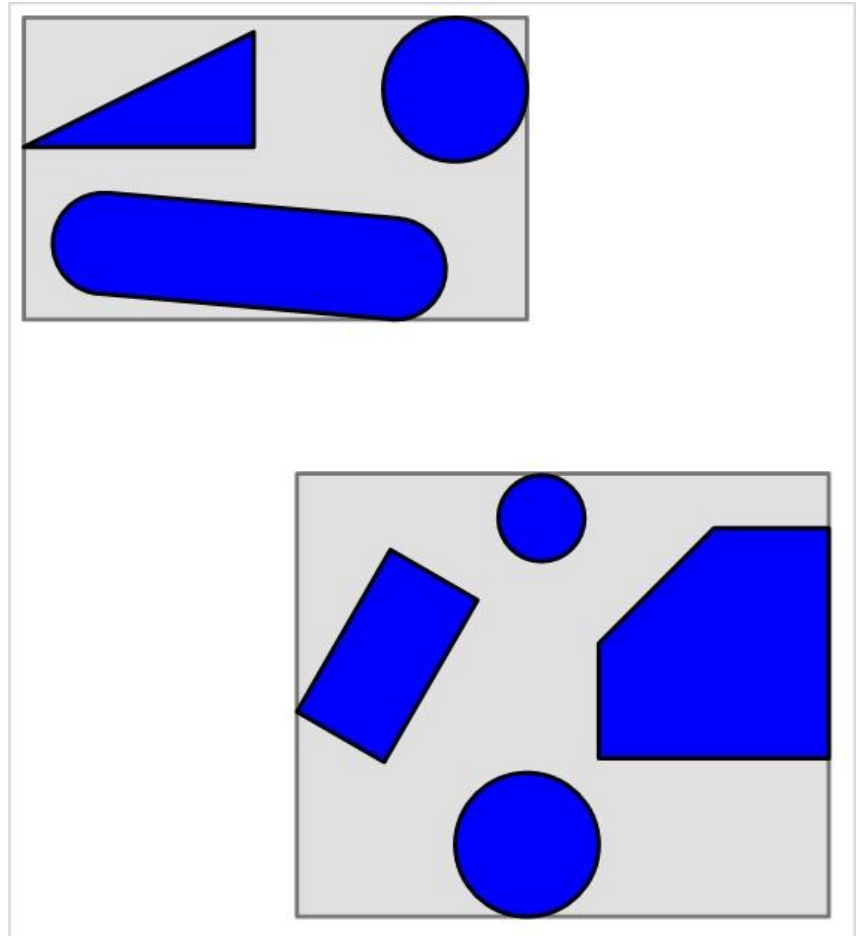
What's the same?

Similar problems:

- Penetration
- Tunneling

Similar techniques:

- Simulation islands



Even 1D Physics is hard

Almost all of the most interesting problems in physics simulation exist even in 1 dimension!

- Distance & separation
- Center of mass
- Attachment
- Penetration
- Tunneling
- Stacking
- Constraints
- Springs and forces
- Gravity
- Simulation Islands
- Tension
- Pressure, density
- Collision detection
- Collision response
- Energy transfer & dampening
- Collision zones / buckets
- Collision pairs
- Contact
- Temporal coherence

Even 1D Physics is hard

Almost all of the most interesting problems in physics simulation exist even in 1 dimension!

- Distance & separation
- Center of mass
- Attachment
- Penetration
- Tunneling
- Stacking
- Constraints
- Springs and forces
- Gravity
- Simulation Islands
- Tension
- Pressure, density
- Collision detection
- Collision response
- Energy transfer & dampening
- Collision zones / buckets
- Collision pairs
- Contact
- Temporal coherence

Differences between 2D and 3D Physics

What's different?

3 main types of differences:

- Some 3D-only constructs are lost completely
- Other concepts are simplified or reduced
- Some new 2D-only algorithms are possible!
- (plus, lots of things are just plain easier, and faster, in 2D)

Apples to ~~Apples~~ Oranges or Apples?

The 2D analogues of 3D constructs are
not as obvious as they might seem.

For instance:

- 3D plane = 2D directed line
- 3D tetrahedron = 2D triangle?
- 3D triangle = 2D triangle?
- 3D sphere = 2D disc
- 3D cylinder = 2D disc OR 2D rect...
- 3D line = 2D line OR 2D point
- 3D volume = 2D area?
- 3D surface area = 2D perimeter?
- 3D distance = 2D distance

Degrees of Freedom

- 3D has 6 Degrees of Freedom (DOF):
 - x, y, z
 - yaw, pitch, roll
- 2D has 3 Degrees of Freedom (DOF):
 - x, y
 - yaw (a.k.a. heading)

3D Orientation

3D orientation is expressible in several ways:

- Euler Angles (yaw, pitch, roll)
 - SUCK BECAUSE: can't interpolate reasonably!
 - SUCK BECAUSE: can't do transformations!
- 3x3 Matrices
 - SUCK BECAUSE: can't interpolate reasonably!
 - SUCK BECAUSE: aren't editable!
- Quaternions (x,y,z,w)
 - SUCK BECAUSE: counter-intuitive!
 - SUCK BECAUSE: aren't human-comprehensible!

3D Orientation



2D Orientation

- 2D orientation is simple: **1 float! (yaw)**



2D Orientation

- 2D orientation is simple: **1 float! (yaw)**
- Occasionally need a 2x2 Matrix to do rotational transforms

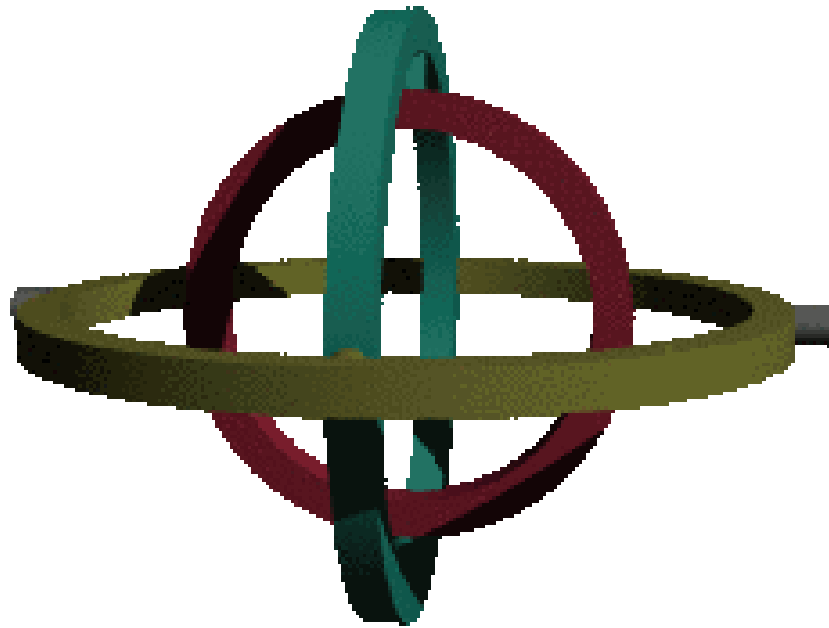
$\cos(Y)$	$\sin(Y)$
$-\sin(Y)$	$\cos(Y)$

2D rotation matrix

$\cos(R)\cos(Y)$ $-\cos(P)\sin(Y)\sin(R)$	$\cos(R)\sin(Y)$ $+\cos(P)\cos(Y)\sin(R)$	$\sin(R)\sin(P)$
$-\sin(R)\cos(Y)$ $-\cos(P)\sin(Y)\cos(R)$	$-\sin(R)\sin(Y)$ $+\cos(P)\cos(Y)\cos(R)$	$\cos(R)\sin(P)$
$\sin(P)\sin(Y)$	$-\sin(P)\cos(Y)$	$\cos(P)$

3D rotation matrix

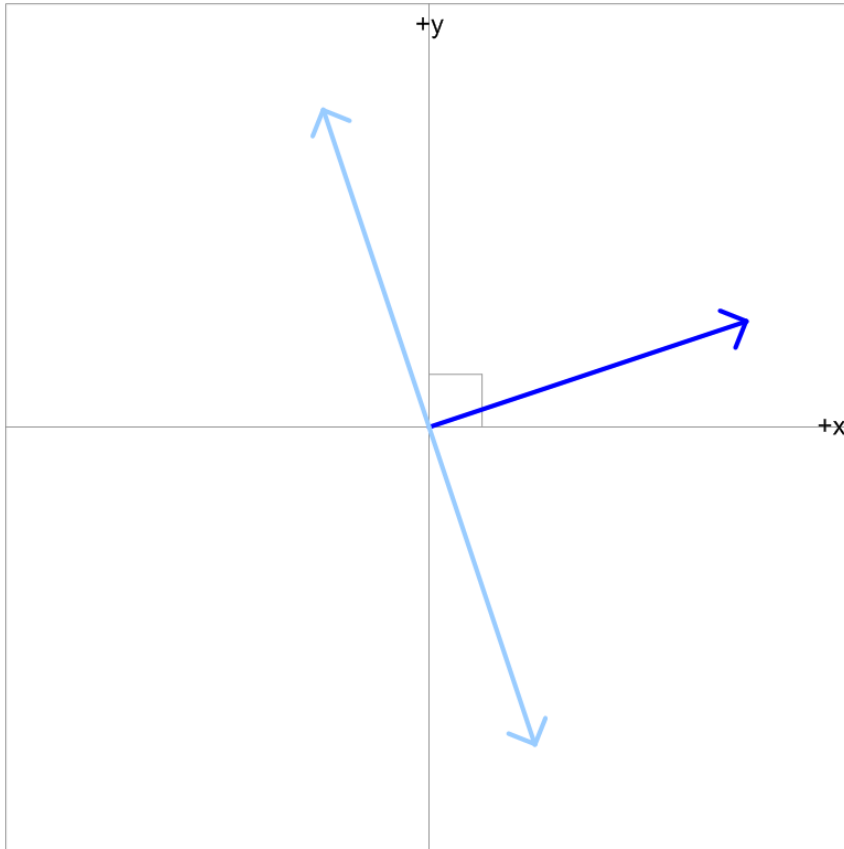
Not in 2D: Gimbal Lock



Not in 2D: Quaternions

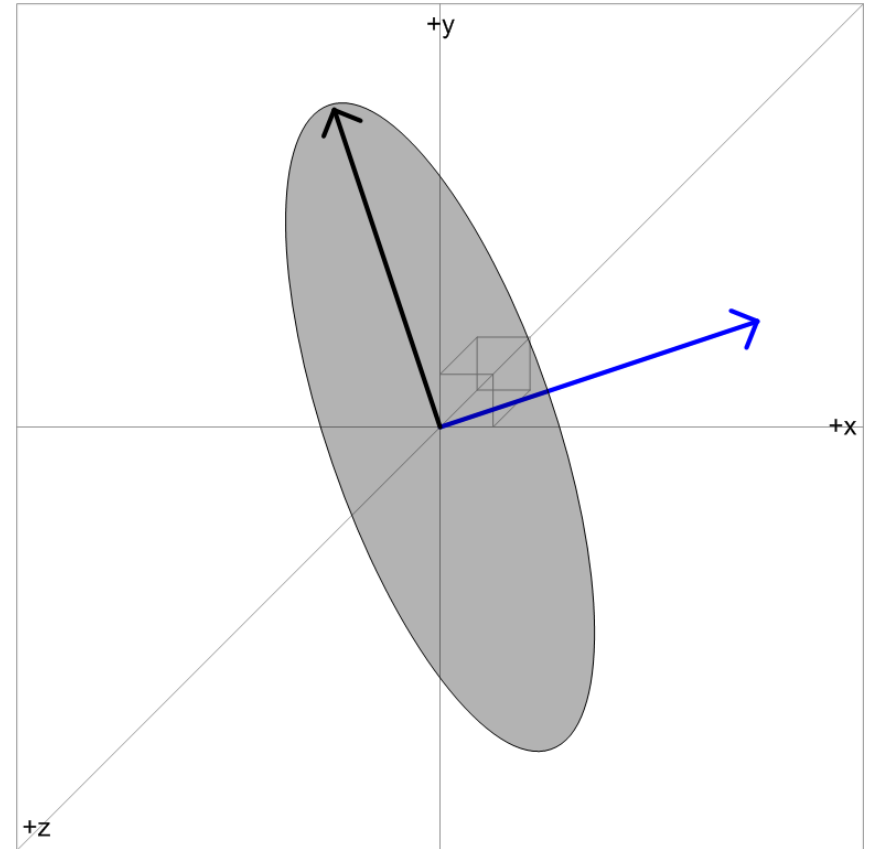
REJOICE!

Perpendicularity



2D Perpendiculars

(only two!)



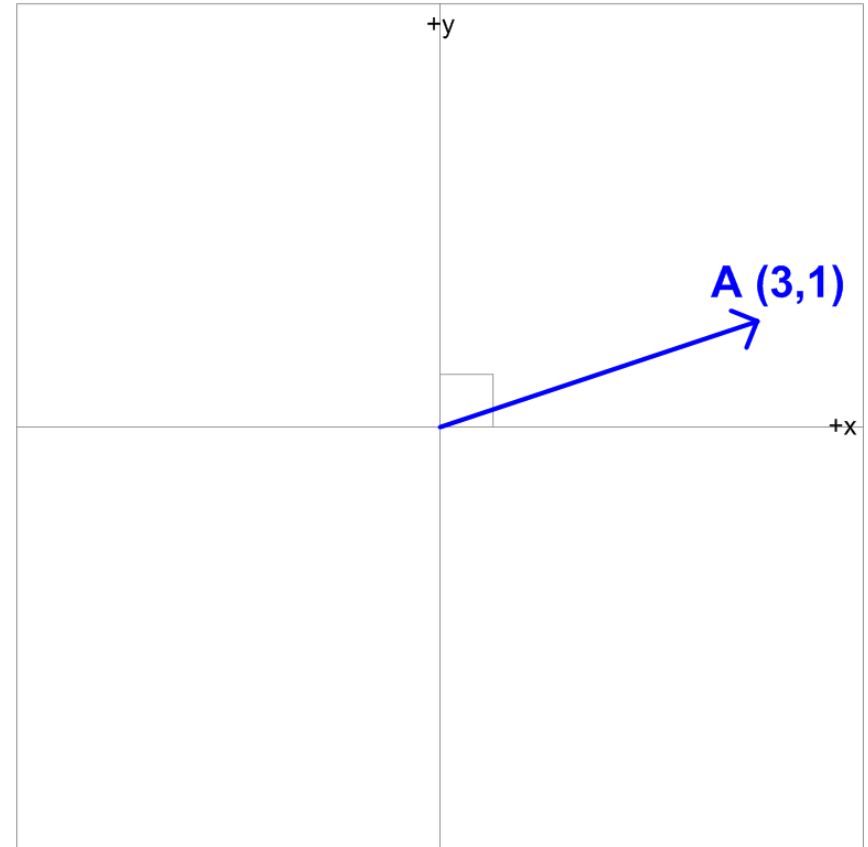
3D Perpendiculars

(infinitely many!)

Perpendicularity

Trivial to compute
perpendiculars in 2D:

$$\mathbf{A} = (A_x, A_y)$$



Perpendicularity

Easy to compute
perpendiculars in 2D:

$$\mathbf{A} = (A_x, A_y)$$

$$\mathbf{A}_{\text{perp1}} = (A_y, -A_x)$$



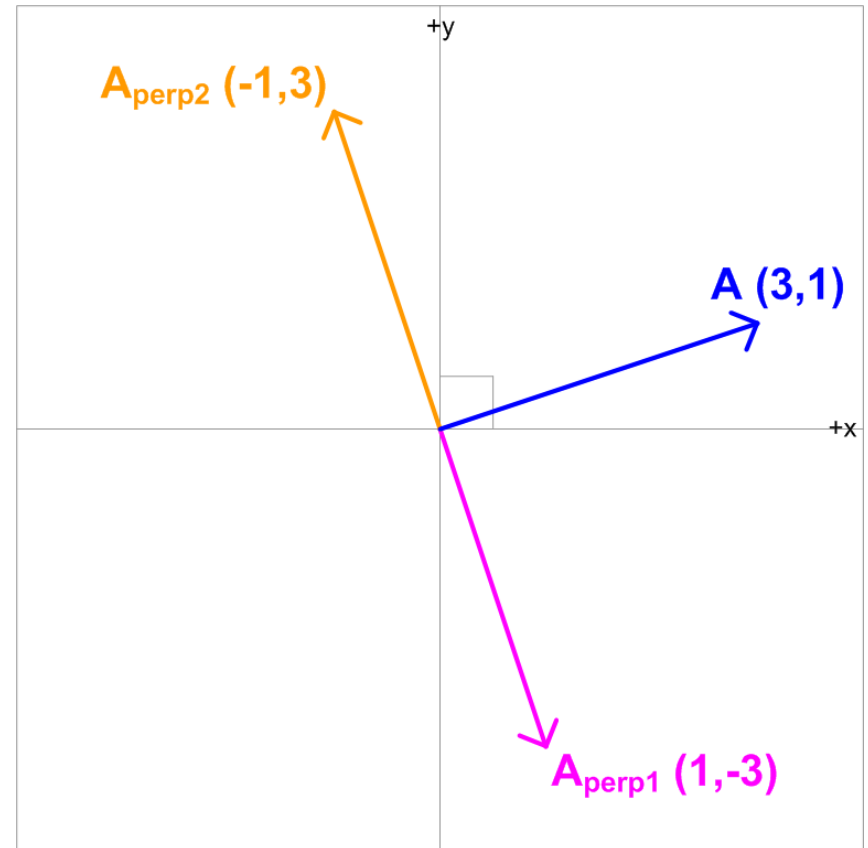
Perpendicularity

Easy to compute
perpendiculars in 2D:

$$\mathbf{A} = (A_x, A_y)$$

$$\mathbf{A}_{\text{perp1}} = (A_y, -A_x)$$

$$\mathbf{A}_{\text{perp2}} = (-A_y, A_x)$$



Not in 2D: ~~Cross Product~~ Vector Cross Product

In 2D, the Cross Product exists, but now it's a **scalar** (float), just like Dot Product.

$$\mathbf{A} \times \mathbf{B} = (A_x B_y - A_y B_x)$$

In fact: the 2D Cross Product $\mathbf{A} \times \mathbf{B}$ is **exactly the same** as a Dot Product with $\mathbf{B}$'s perpendicular:

$$\mathbf{A} \times \mathbf{B} = \mathbf{A} \bullet \mathbf{B}_{\text{perpendicular}}$$

2D Data is smaller

Size Comparisons

2D

Position, Velocity,
Acceleration

$(x,y) = 2$ floats

Orientation

$(\text{yaw}) = 1$ float

$(\text{Matrix}22) = 4$ floats

3D

Position, Velocity,
Acceleration

$(x,y,z) = 3$ floats

Orientation

$(\text{Euler}) = 3$ floats

$(\text{Matrix}33) = 9$ floats

$(\text{Quaternion}) = 4$ floats

Size Comparisons

2D

Angular Velocity

1 float

Mass

1 float (mass)

3D

Angular Velocity

3 floats (Vector3)

Mass

1 float (mass) +
9 floats (inertial
tensor matrix)

Size Comparisons

2D

AABB (axis-aligned
bounding box)

4 floats (mins, maxs)

OBB (oriented bounding
box)

4-8 floats

3D

AABB (axis-aligned
bounding box)

9 floats (mins, maxs)

OBB (oriented bounding
box)

9-12 floats

Size Comparisons

In general, 3D objects have roughly 2 to 3 times as many features as their 2D counterparts do.

Which means that N^2 types of algorithms (like discrete / explicit CSO generation?) can sometimes take anywhere from 4x to 9x as long in 3D!

2D Computations are faster

Speed Comparisons

2D

Dot Product

2 mults + 1 add

Scalar Cross Product

2 mults + 1 subtract

3D

Dot Product

3 mults + 2 adds

Vector Cross Product

6 mults + 3 subtracts

Speed Comparisons

2D

Create Rotation Matrix

1 sine + 1 cosine

Apply Matrix Transform
to Vector

2 Dot Products

(4 mults + 2 adds)

3D

Create Rotation Matrix

6 sines/cosines +
16 mults + 4 adds

Apply Matrix Transform
to Vector

3 Dot Products

(9 mults + 6 adds)

Speed Comparisons

2D

Concatenate Rotations

1 add (float + float!)

Separating Axis Test (SAT)
vs. OBB

32 Dot Products

(64 mults + 32 adds)

3D

Concatenate Rotations

9 Dot Products

(27 mults + 18 adds)

Separating Axis Test (SAT)
vs. OBB

9 Cross Products

120 Dot Products

(414 mults + 267 adds)

Speed Comparisons

2D

BSP Traversal

LgN 2D Dot Products
& comparisons

Sweep and Prune

2/3 as much as 3D

3D

BSP Traversal

LgN 3D Dot Products &
comparisons

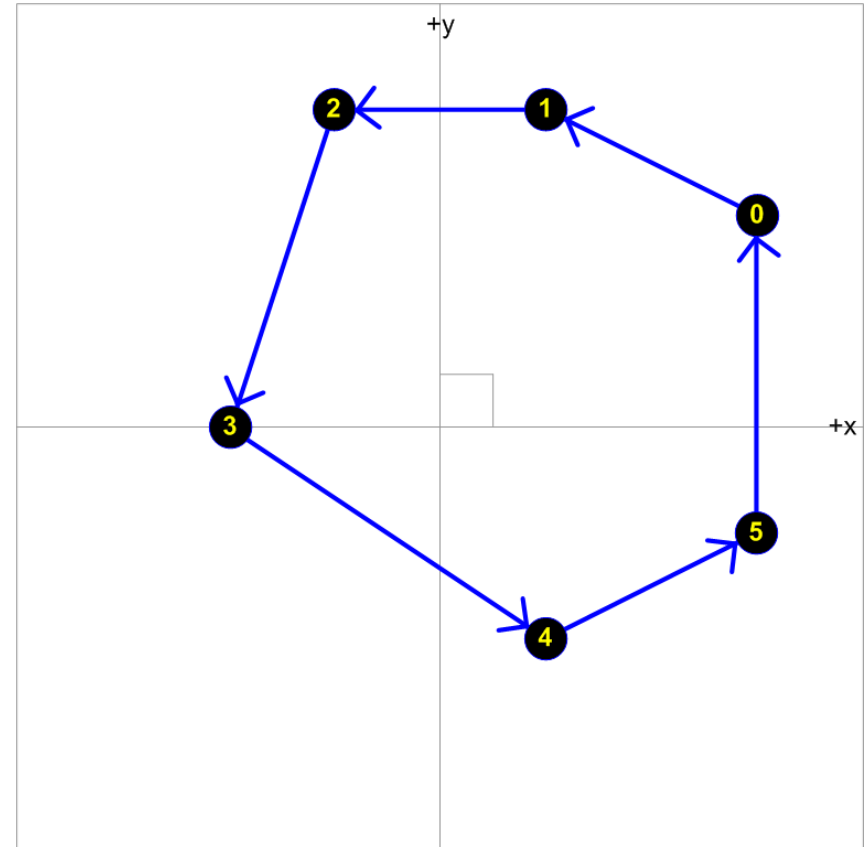
Sweep and Prune

50% more than 2D

Only in 2D: Ordered Features

It is common to represent 2d objects as a CCW “windings” of points and/or features (edge faces, etc.)

Useful for LOTS of algorithms!



2D SIMD “Parallelism”

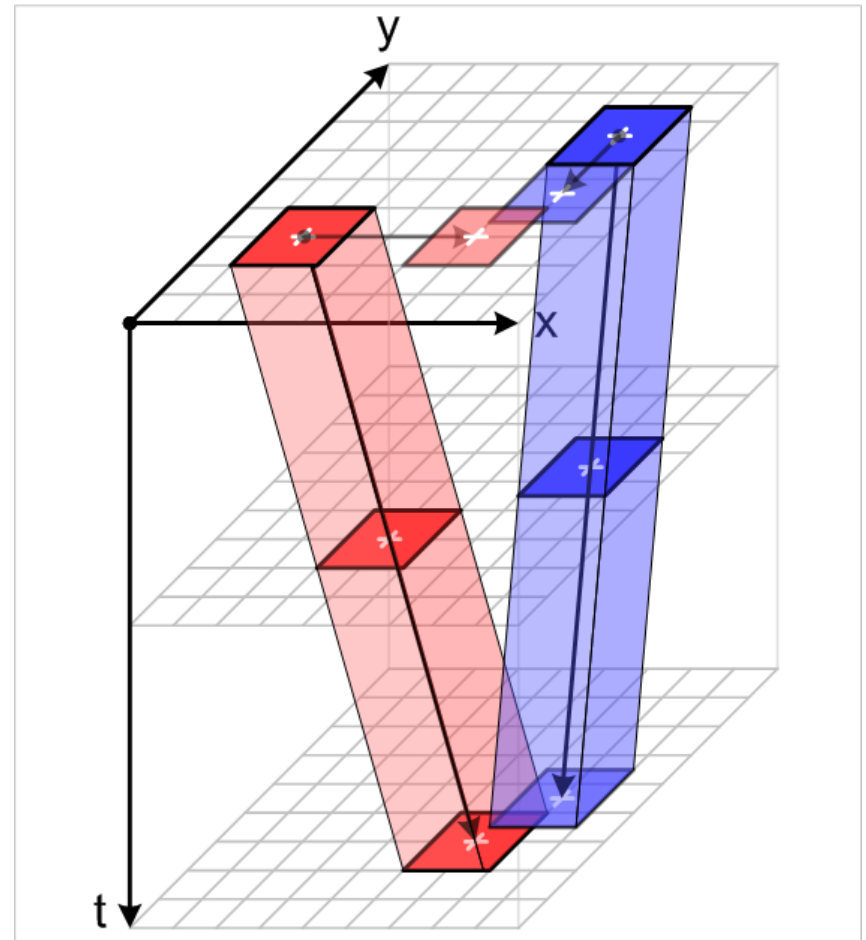
- Can use SIMD operations to do “two at once”
pipelining with pairs of Vector2s in many cases.
Extreme example:

$$\begin{aligned} & (Ax, Ay, Bx, By) * (Qx, Qy, Qx, Qy) \\ & (Cx, Cy, Dx, Dy) * (Rx, Ry, Rx, Ry) \\ & (AQx, BQx, CRx, DRx) + (AQy, BQy, C Ry, DRy) \end{aligned}$$

= $A \bullet Q$, $B \bullet Q$, $C \bullet R$, $D \bullet R$ in 3 (2D) SIMD instructions
instead of 20 (3D) float instructions

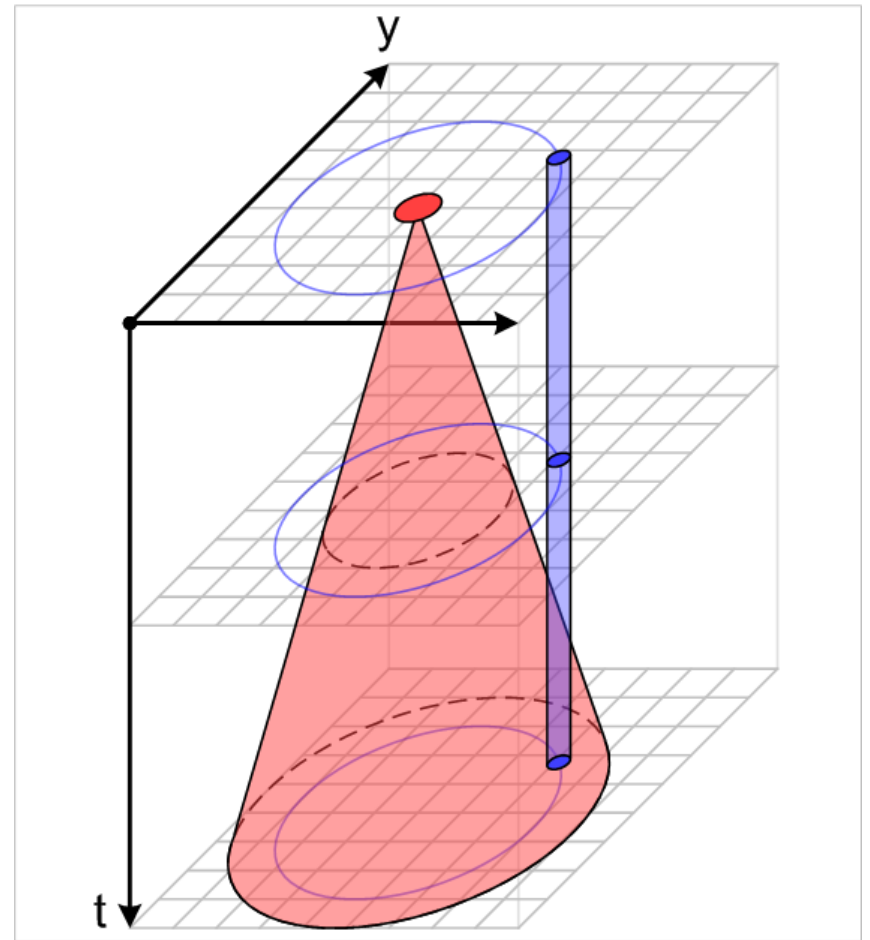
2D Spacetime is 3D!

- If you need to use spacetime (for continuous collision detection checks or whatnot), in 2d you simply go to 3d spacetime (x,y,t) . Yay! we can still pretty much picture this in our heads.
- In 3d, if you need to go spacetime you end up in 4d (x,y,z,t) . Boo!
 - This is very hard to visualize (but doable)
 - might not have the 4d functions already in your math library



2D Spacetime is 3D!

- If you need to use spacetime (for continuous collision detection checks or whatnot), in 2d you simply go to 3d spacetime (x,y,t) . Yay! we can still pretty much picture this in our heads.
- In 3d, if you need to go spacetime you end up in 4d (x,y,z,t) . Boo!
 - This is very hard to visualize (but doable)
 - might not have the 4d functions already in your math library



2D Bit Bucket Tests

- It is much more reasonable to divide a 2d space into reasonable buckets for doing tricks like Bit Bucket Tests:
 - Chop space into N (32, or 64, or 96, or 128) different areas
 - Each object has N bits which are updated when it moves. A bit is lit if the object is (at all) inside the corresponding area
 - Object vs. Object early rejection test: MyBits & YourBits
 - Note: I used this on a project recently and was able to reduce collision detection time by 90%!

The Best Thing About 2D is...

- Easier to draw (paper / whiteboard), so...
- Easier to get it right, so...
- Easier to avoid bugs, AND:
- Easier to learn new algorithms, which means...
- Easier to implement, and...
- Simpler code (in some cases), which gives you...
- Easier to debug, all of which...
- Gets you to coding the next cool thing (physics feature, or *gasp* gameplay) that much sooner!

Remember:

- If you're doing 3D physics in general – or some piece of 3D math in particular – ask yourself if you could be doing (some, or all of) it in 2D instead.
- If you're already doing 2D, realize that certain aspects of algorithms and code you find in whitepapers & books & online may be able to be simplified dramatically.
- Look for opportunities to take advantage of the wins that 2D brings you!

Thanks!

Feel free to email me at:

Squirrel@TrueThought.com