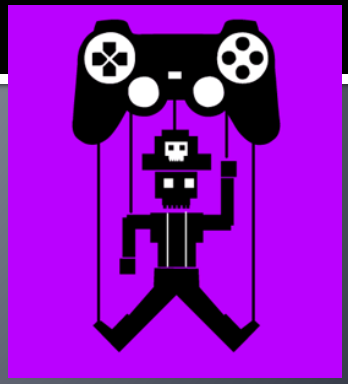
Faculty of Mathematics and Physics
Charles University, Prague
3rd October 2024

Gameplay Programming Level 01

What is GPP anyway?

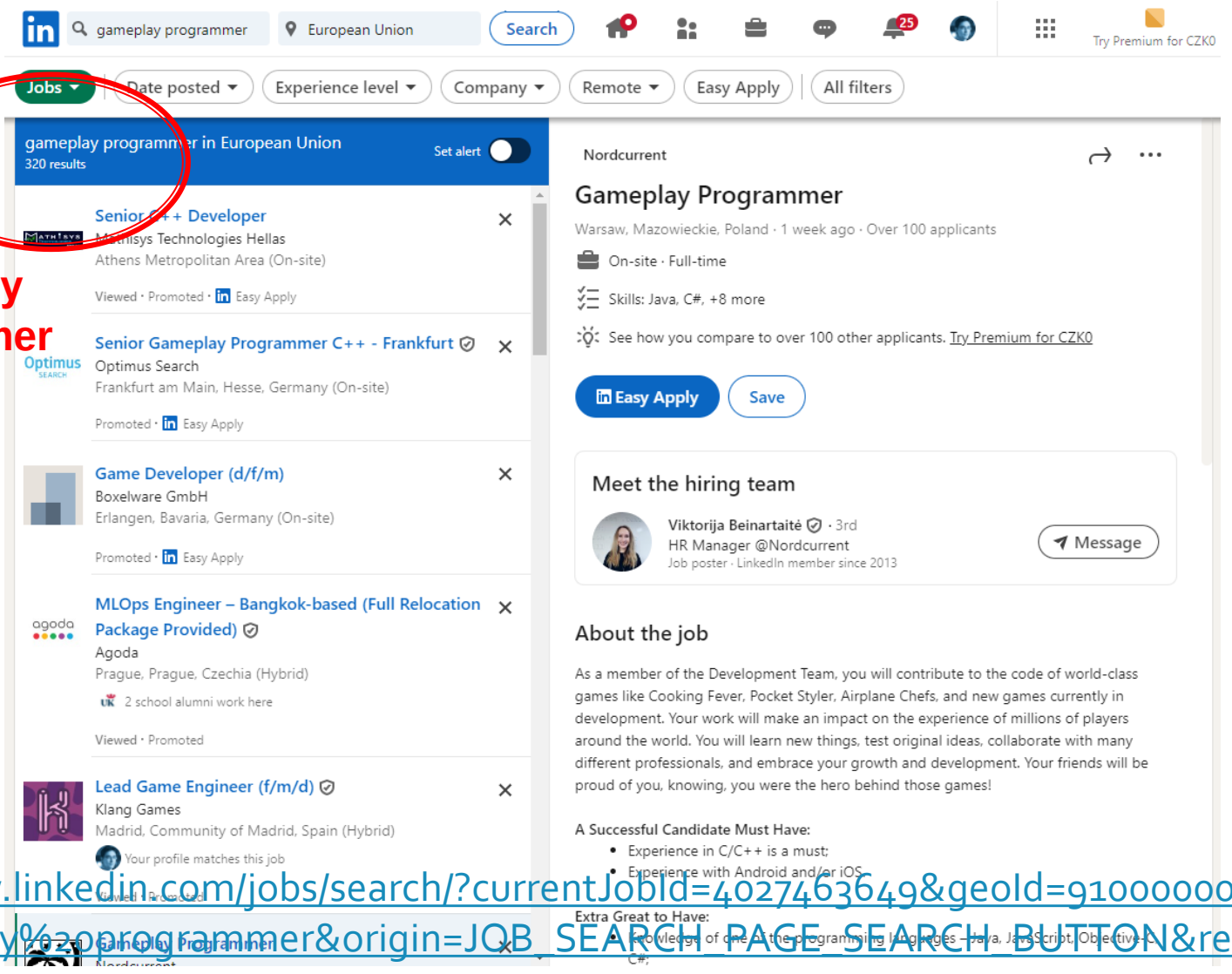
An introduction (albeit a confusing one…)*



* You have been warned.

Gameplay Programming

What is… ?



“Gameplay Programmer in EU”

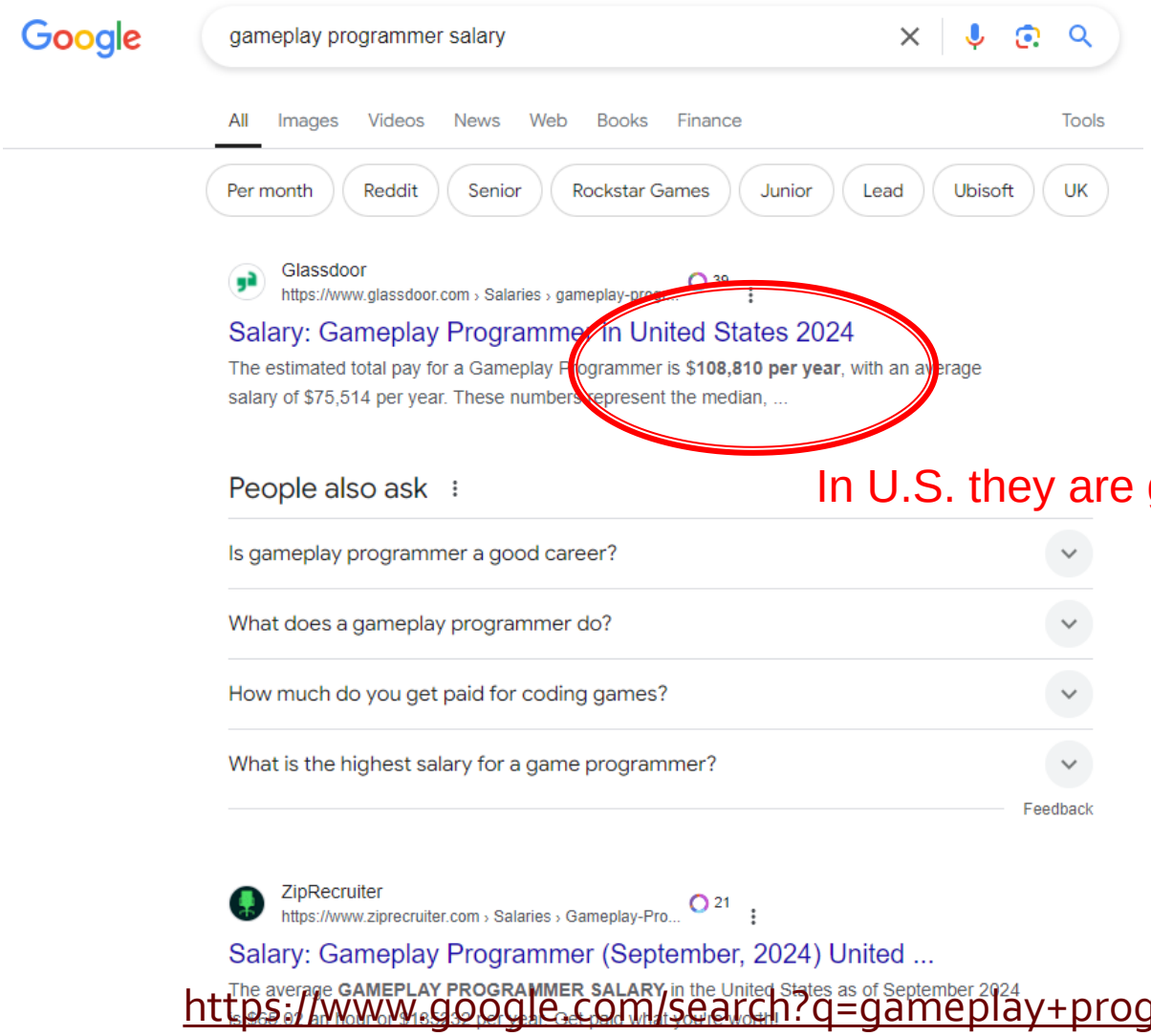
…

they exist

https://www.linkedin.com/jobs/search/?currentJobId=4027463649&geoId=91000000&keywords=gameplay%20programmer&origin=JOB_SEARCH_PAGE_SEARCH_BUTTON&refresh=true

Gameplay Programming

What is… ?

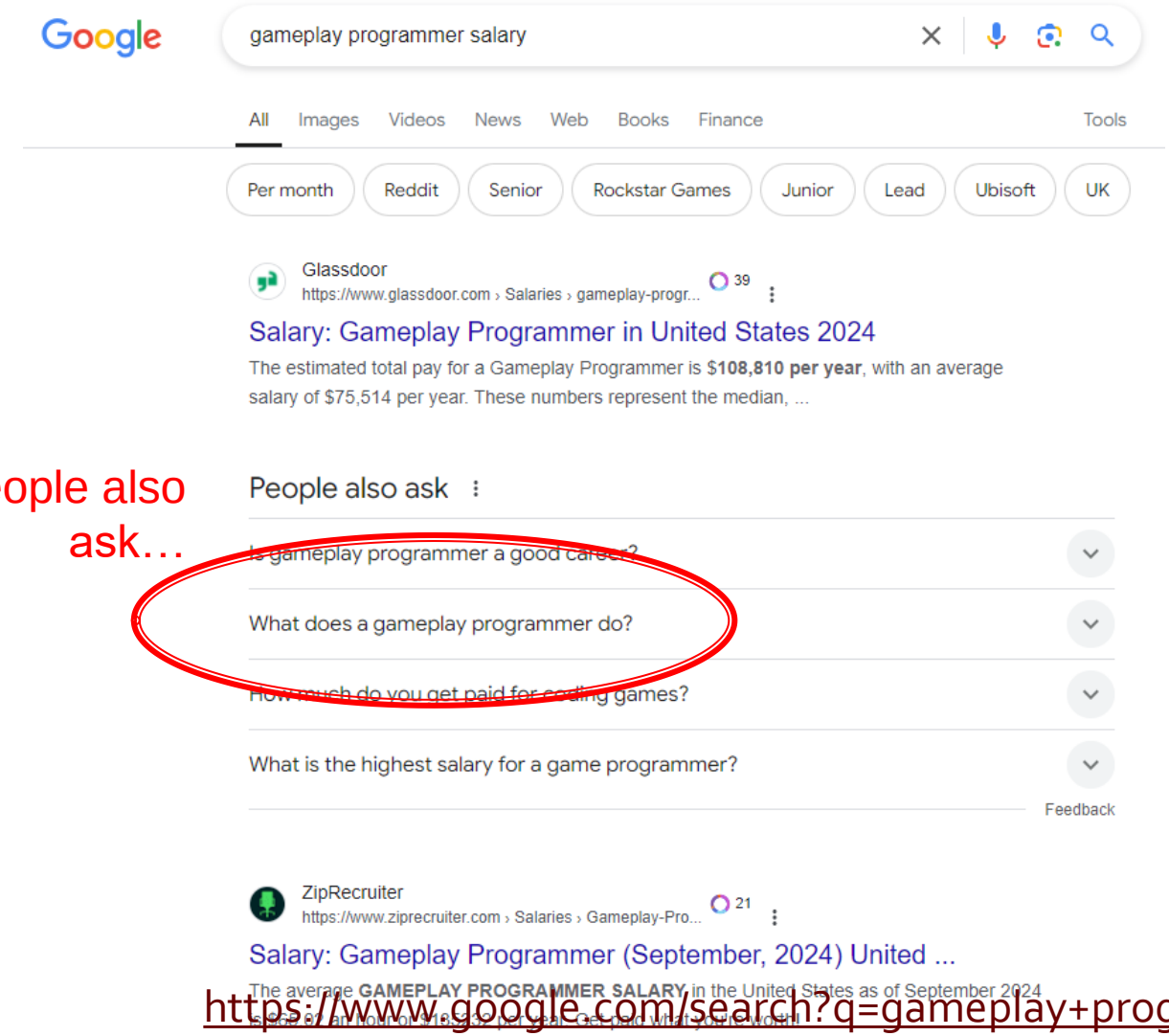


In U.S. they are getting paid

https://www.google.com/search?q=gameplay+programmer+salary

Gameplay Programming

What is… ?



But people also ask…

https://www.google.com/search?q=gameplay+programmer+salary

Gameplay Programming

What is… ?

Google

gameplay programmer

Vše | Obrázky | Nákupy | Videa | Zprávy | Více | Nastavení | Nástroje

Přibližný počet výsledků: 4 770 000 (0,32 s)

And there seems to be a definition of them…

Gameplay programmers write the code for the interactions that make a game fun to play. While lead designers decide on the combat, **gameplay programmers** make it happen. They work with level designers to see what needs to be done to make the **gameplay** work.

www.screenskills.com › careers › games › programming

Gameplay programmer - ScreenSkills

O vybraných úryvcích | Zpětná vazba

https://www.google.com/search?q=gameplay+programmer

Gameplay Programming

What is… ?

Google

gameplay programmer

Vše Obrázky Nákupy Videa Zprávy Více Nastavení Nástroje

Přibližný počet výsledků: 4 770 000 (0,32 s)

Without a player, there is no combat…

Gameplay programmers write the code for the interactions that make a game fun to play. While lead designers decide on the combat, **gameplay programmers** make it happen. They work with level designers to see what needs to be done to make the **gameplay** work.

www.screenskills.com › careers › games › programming

Gameplay programmer - ScreenSkills

O vybraných úryvcích Zpětná vazba

https://www.google.com/search?q=gameplay+programmer

Gameplay Programming

What is… ?

Google

gameplay programmer

Vše Obrázky Nákupy Videa Zprávy Více Nastavení Nástroje

Přibližný počet výsledků: 4 770 000 (0,32 s)

Gameplay programmers write the code for the interactions that make a game fun to play. While lead designers decide on the combat, **gameplay programmers** make it happen. They work with level designers to see what needs to be done to make the **gameplay** work.

www.screenskills.com › careers › games › programming

Gameplay programmer - ScreenSkills

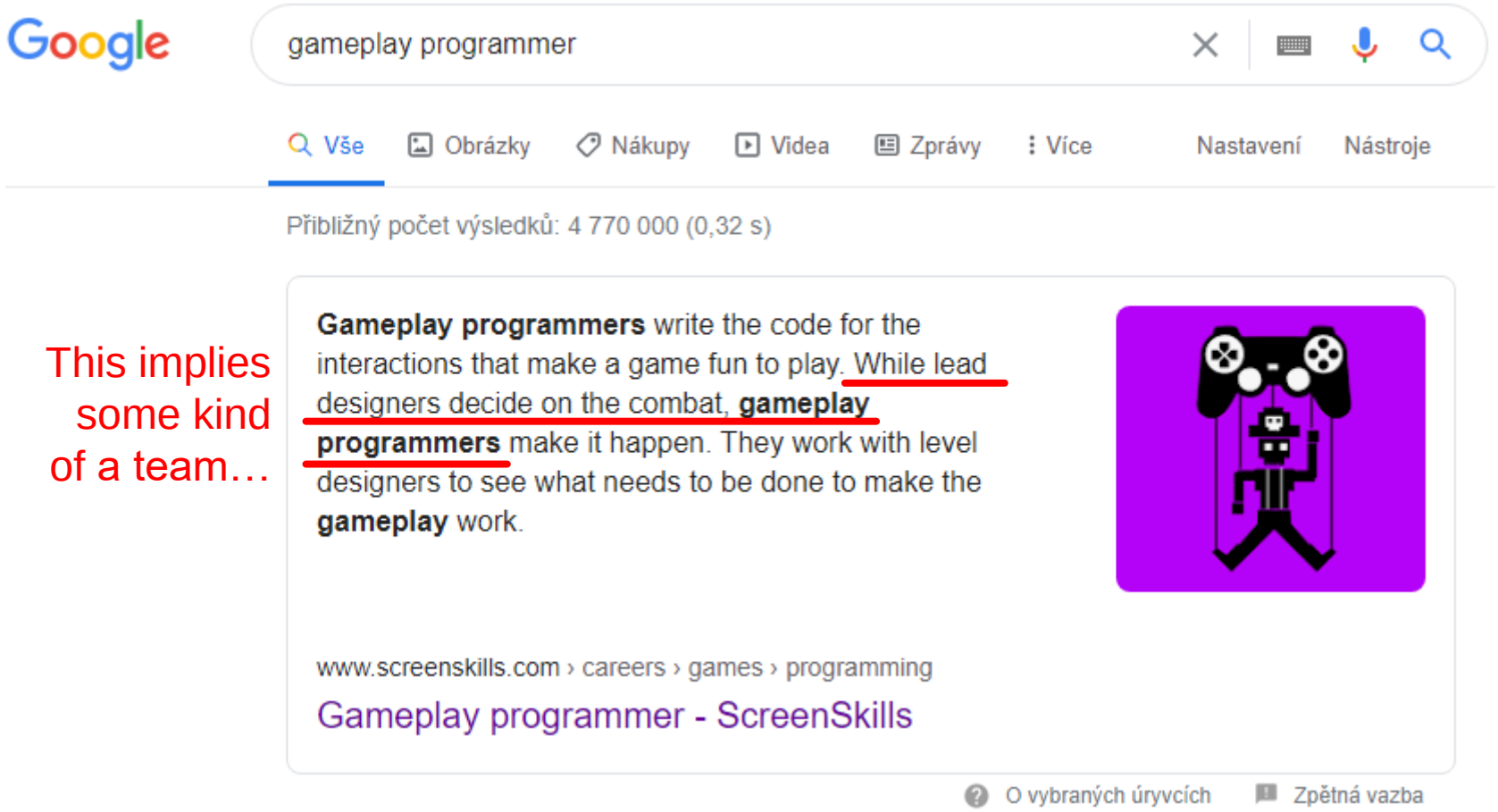
O vybraných úryvcích Zpětná vazba

Who would have guessed…

https://www.google.com/search?q=gameplay+programmer

Gameplay Programming

What is… ?

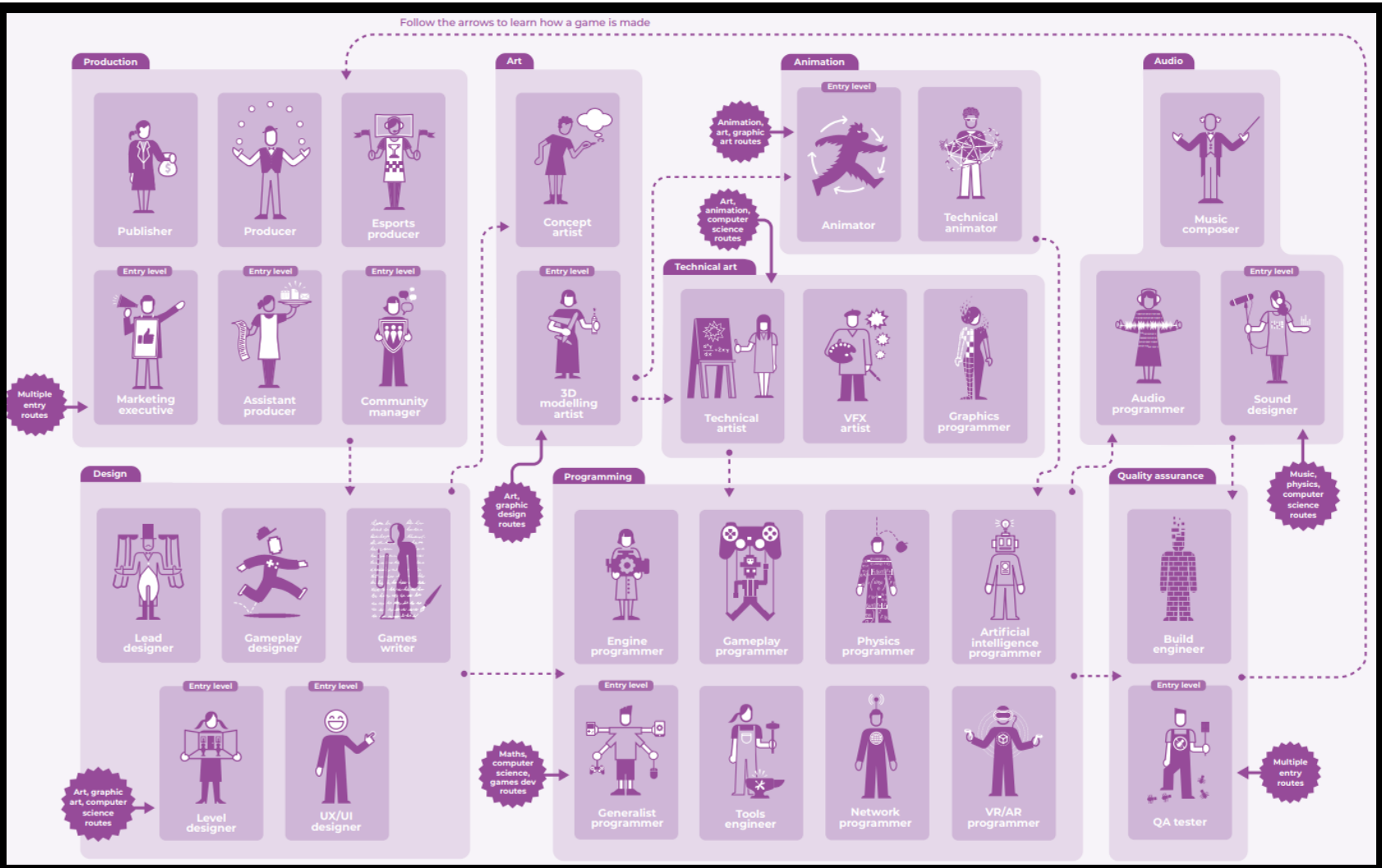


This implies some kind of a team…

https://www.google.com/search?q=gameplay+programmer

Gameplay Programming

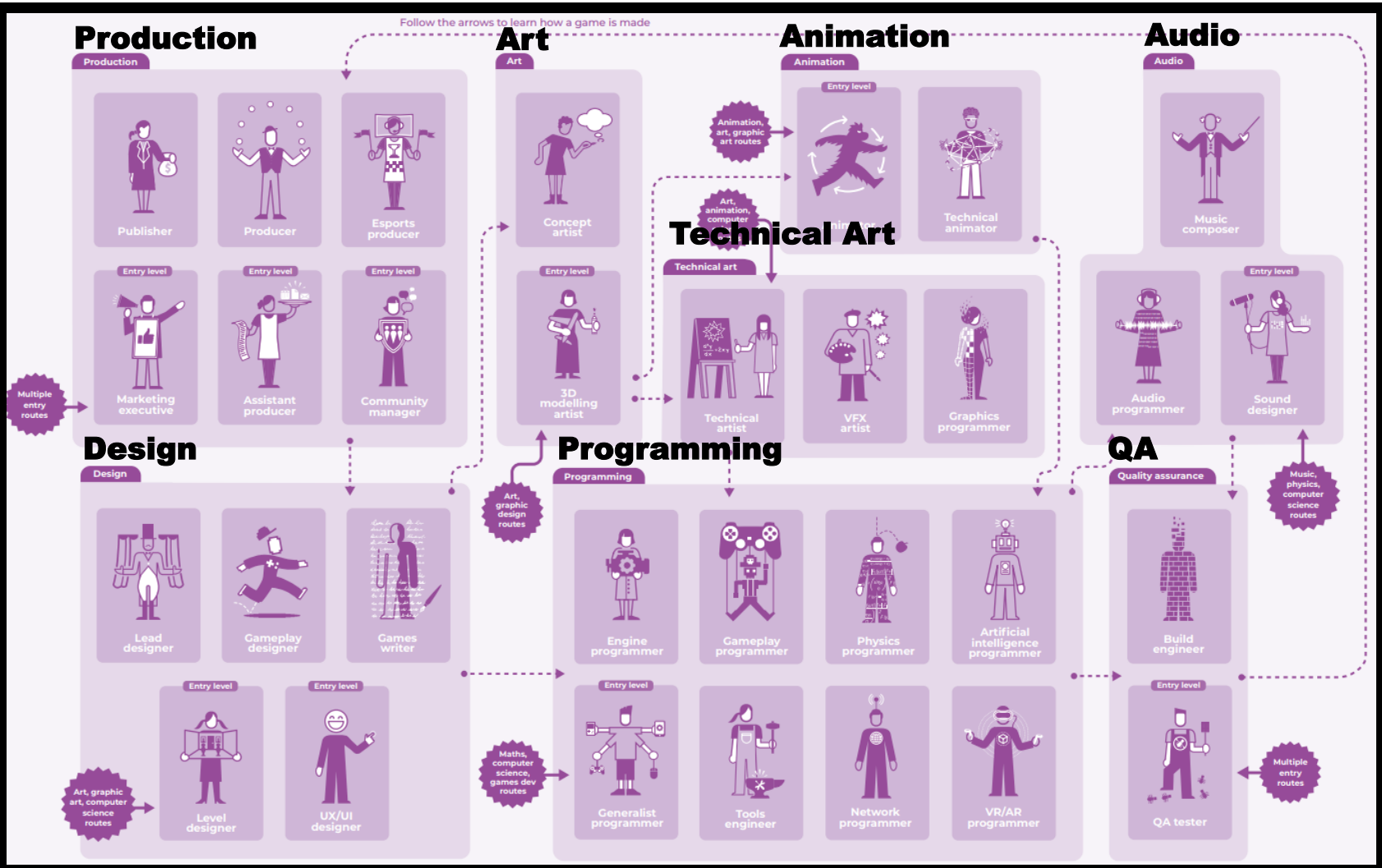
What is… ?



https://www.screenskills.com/media/2595/games-map-downloadable-final.pdf

Gameplay Programming

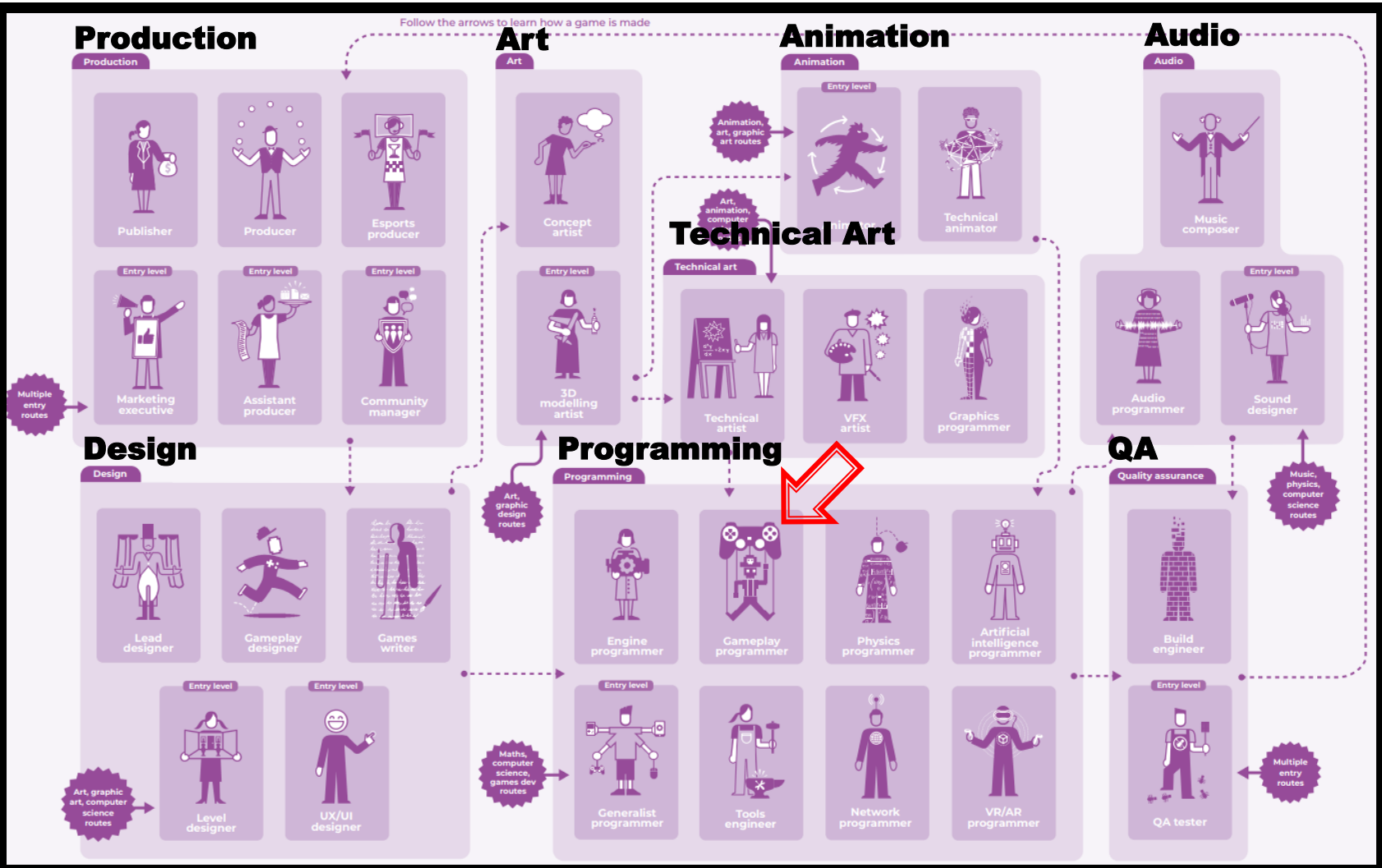
What is… ?



https://www.screenskills.com/media/2595/games-map-downloadable-final.pdf

Gameplay Programming

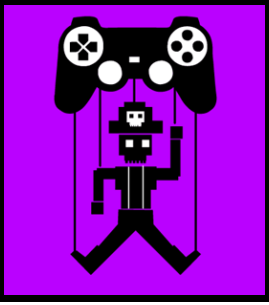
What is… ?

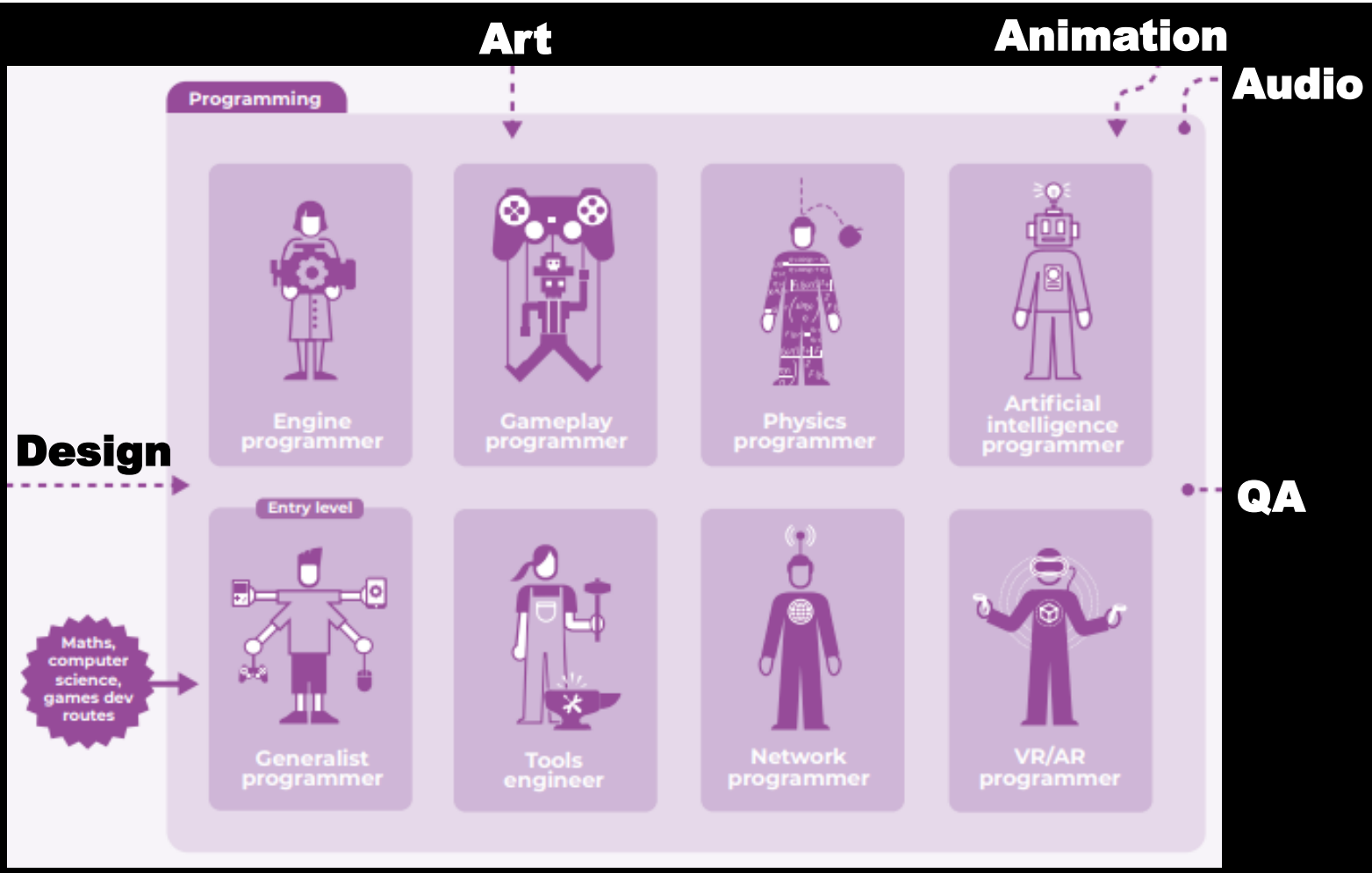


https://www.screenskills.com/media/2595/games-map-downloadable-final.pdf

Gameplay Programming

What is… ?

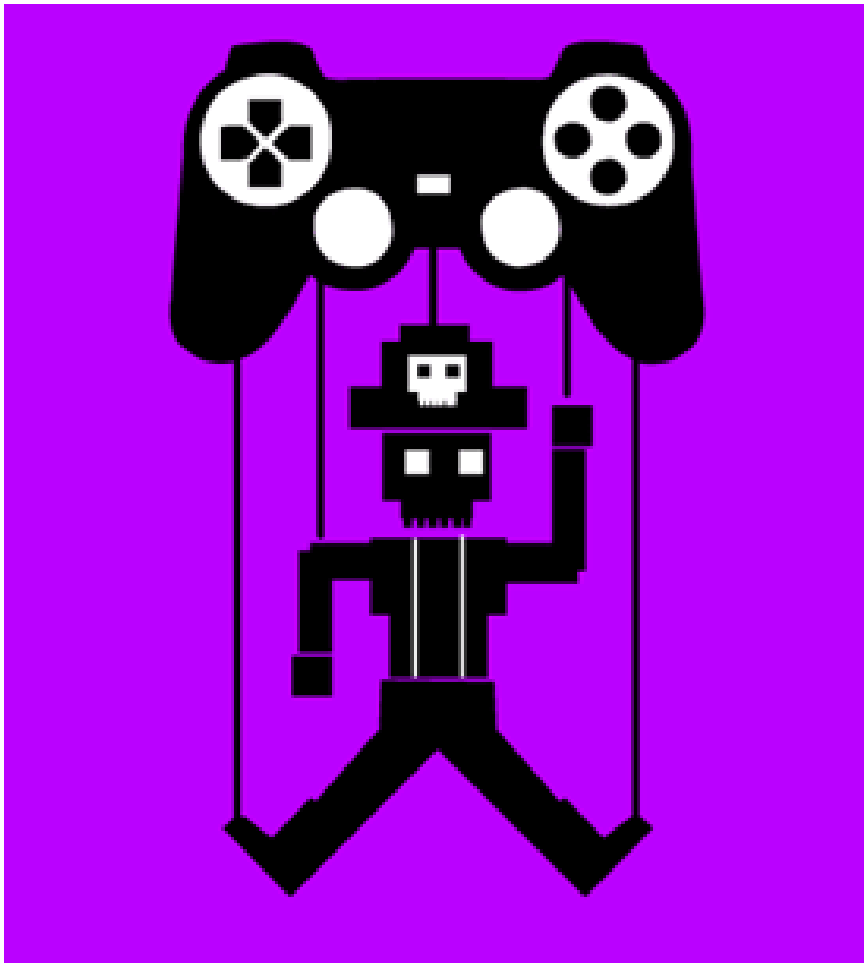




https://www.screenskills.com/media/2595/games-map-downloadable-final.pdf

Gameplay Programming

What is… ?



Gameplay Programming at Ubisoft

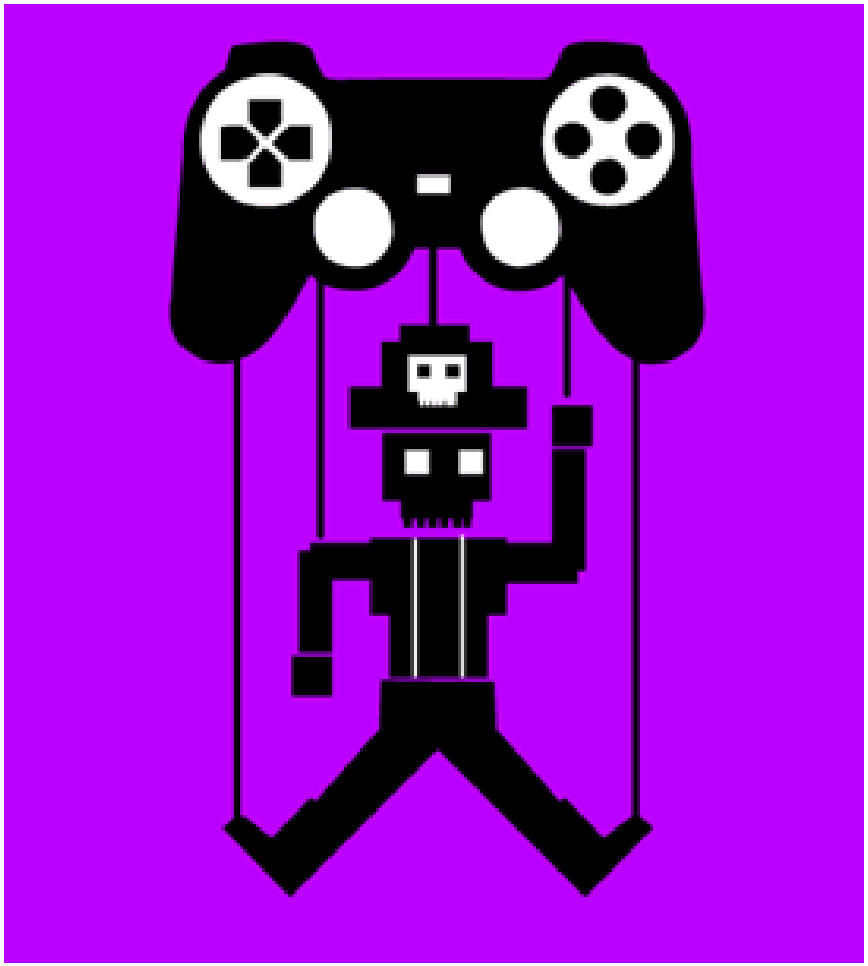
"It's kind of a physical art. It is about trying to manipulate that interface between the player and the game."

https://www.youtube.com/watch?v=PNpxtl8KReg

[3:15]

Gameplay Programming

What is… ?



Gameplay Programming at Ubisoft

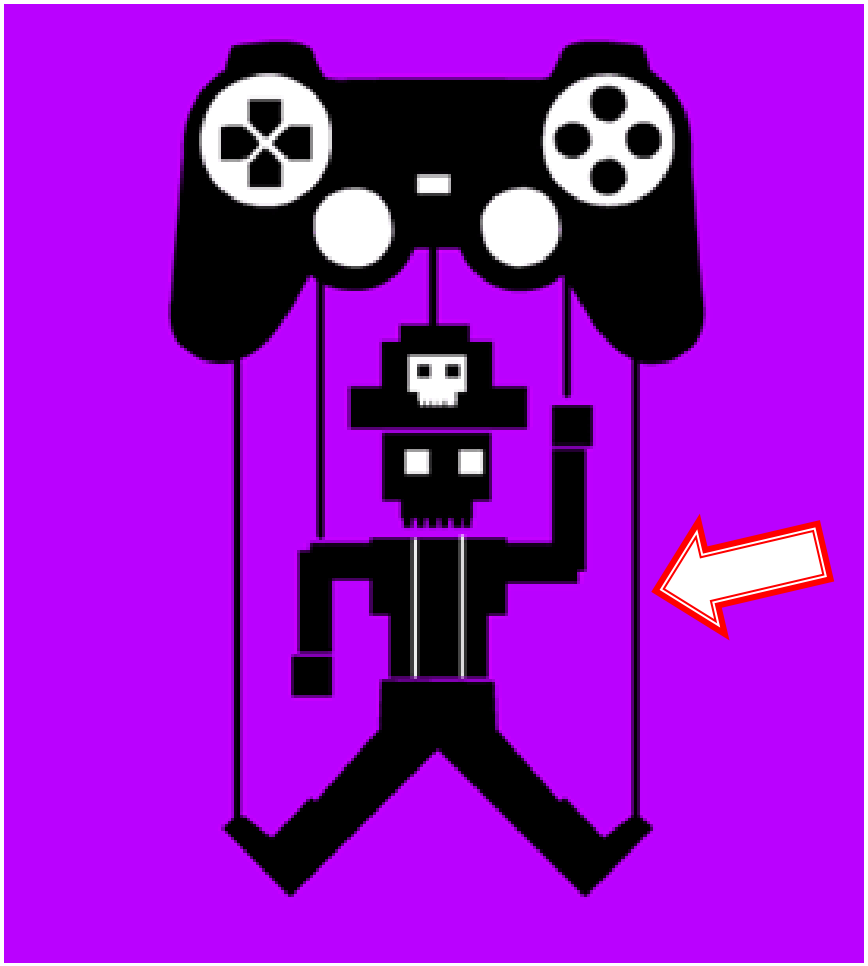
"It's kind of a physical art. It is about trying to manipulate that interface between the player and the game."

https://www.youtube.com/watch?v=PNpxtl8KReg

[3:15]

Gameplay Programming

What is… ?



Gameplay Programming at Ubisoft

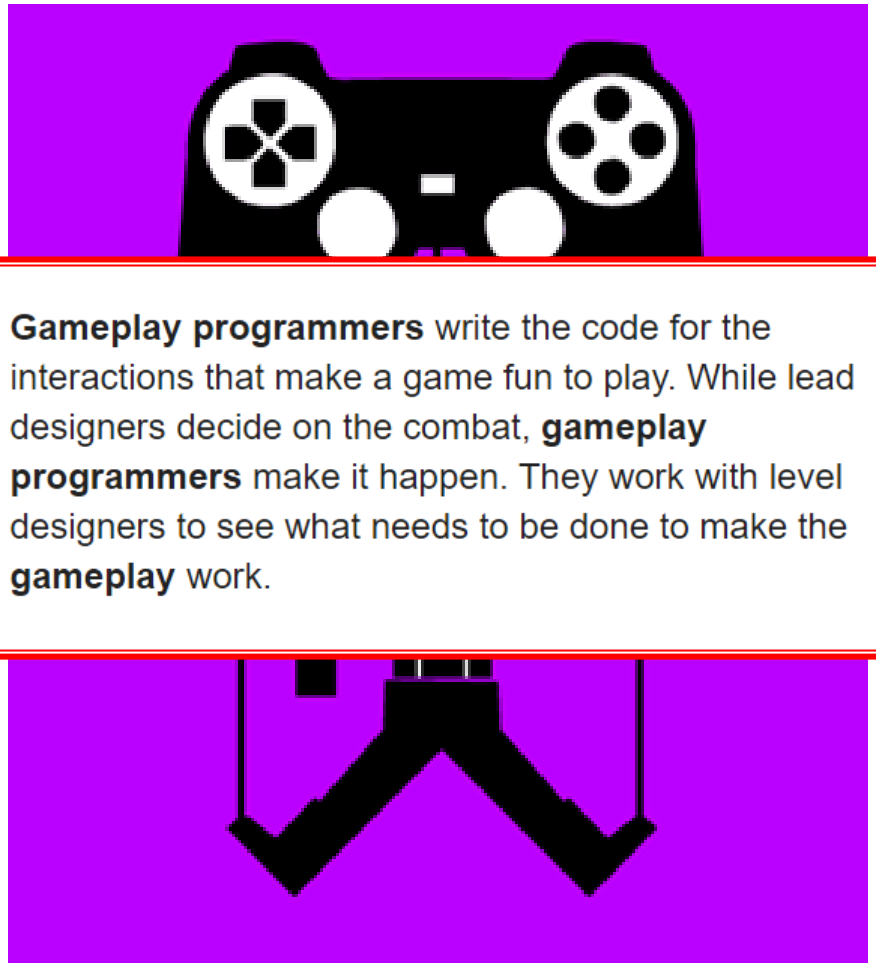
"It's kind of a physical art. It is about trying to manipulate that interface between the player and the game."

https://www.youtube.com/watch?v=PNpxtl8KReg

[3:15]

Gameplay Programming

What is… ?



Gameplay programmers write the code for the interactions that make a game fun to play. While lead designers decide on the combat, **gameplay programmers** make it happen. They work with level designers to see what needs to be done to make the **gameplay** work.

Gameplay Programming at Ubisoft

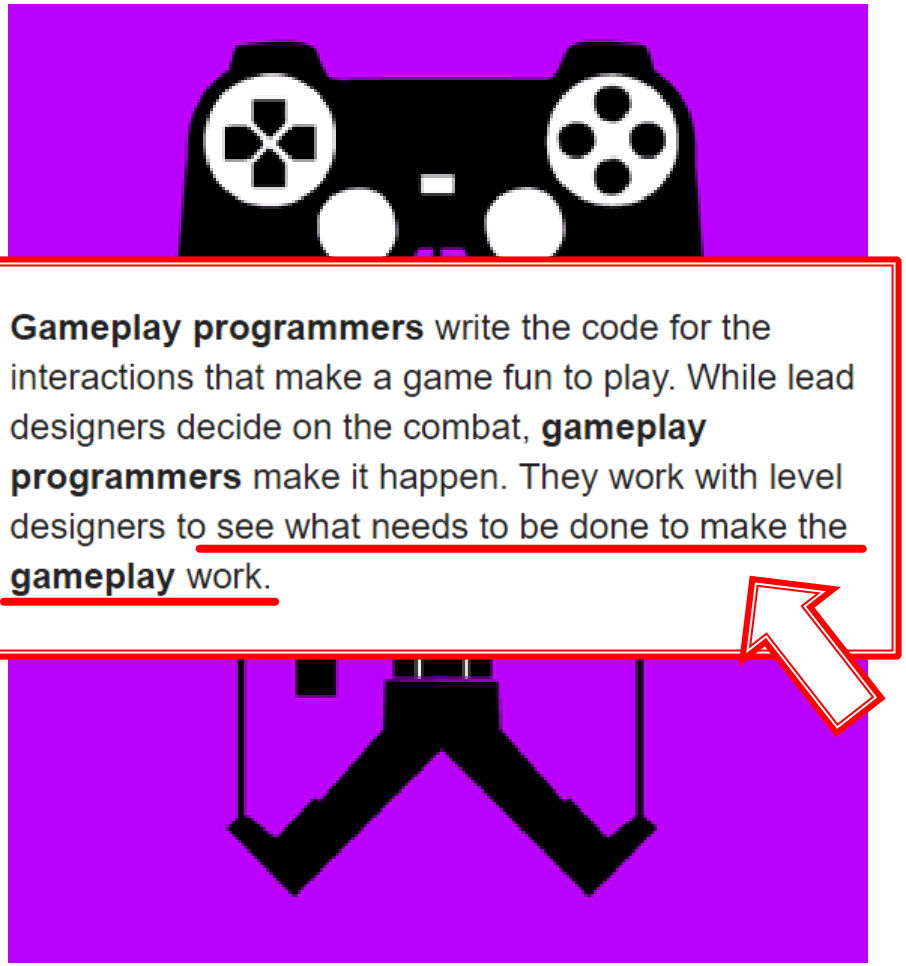
"It's kind of a physical art. It is about trying to manipulate that interface between the player and the game."

https://www.youtube.com/watch?v=PNpxtl8KReg

[3:15]

Gameplay Programming

What is… ?



Gameplay Programming at Ubisoft

"It's kind of a physical art. It is about trying to manipulate that interface between the player and the game."

https://www.youtube.com/watch?v=PNpxtl8KReg

[3:15]

Gameplay Programming

What is… ?

Gameplay

Gameplay Programming

What is… ?

Gameplay

https://en.wikipedia.org/wiki/Gameplay

Gameplay is the specific way in which players interact with a game, and in particular with video games. Gameplay is the pattern defined through the game rules, connection between a player and the game, challenges and overcoming them, plot and player's connection with it. Video game gameplay is distinct from graphics and audio elements.

Gameplay Programming

What is… ?

Gameplay

https://en.wikipedia.org/wiki/Gameplay

Gameplay is the specific way in which players interact with a game, and in particular with video games. Gameplay is the pattern defined through the game rules, connection between a player and the game, challenges and overcoming them, plot and player's connection with it. Video game gameplay is distinct from graphics and audio elements.

Gameplay Programming

What is… ?

Gameplay

https://en.wikipedia.org/wiki/Gameplay

Gameplay is the specific way in which players interact with a game, and in particular with video games. Gameplay is the pattern defined through the game rules, connection between a player and the game, challenges and overcoming them, plot and player's connection with it. Video game gameplay is distinct from graphics and audio elements.

Gameplay Programming
What is… ?

Gameplay (as an ambiguous term)
https://en.wikipedia.org/wiki/Gameplay

"A series of interesting choices." *Sid Meier* [13]

"The structures of player interaction with the game system and with other players in the game." *Björk, Holopainen* [14]

"One or more causally linked series of challenges in a simulated environment." *Adams, Rollings* [15]

"Gameplay here is seen as the interactive gaming process of the player with the game." *Nacke et. al.* [17]

Gameplay Programming
What is… ?

Gameplay (as an ambiguous term)
https://en.wikipedia.org/wiki/Gameplay

"A series of interesting choices." *Sid Meier* [13]

"The structures of player interaction with the game system and with other players in the game." *Björk, Holopainen* [14]

"One or more causally linked series of challenges in a simulated environment." *Adams, Rollings* [15]

"Gameplay here is seen as the interactive gaming process of the player with the game." Nacke et. al. [17]

"The **experience** of **gameplay** is one of interacting with a game design in the performance of cognitive tasks, with a variety of emotions arising from or associated with different elements of motivation, task performance and completion." *Lindley, Nacke, Sennersten* [3]

Gameplay Programming
What is… ?

Gameplay (as an ambiguous term)
https://en.wikipedia.org/wiki/Gameplay

"The **experience** of **gameplay** is one of interacting with a game design in the performance of cognitive tasks, with a variety of emotions arising from or associated with different elements of motivation, task performance and completion."
Lindley, Nacke, Sennersten [3]

Quality of gameplay is often associated with playability.

Playability is the ease by which the game can be played or the quantity or duration that a game can be played.

Playability evaluative methods target games to improve design while player experience evaluative methods target players to improve gaming. *Nacke et al.* [17]

Gameplay programmer is here to assure good playability.

Gameplay Programming
What is… ?

Playability

Playability is defined as: a set of properties that describe the player experience using a specific game system… characterized by different attributes and properties to measure the video game player experience. *González Sánchez et al.* [19]

Satisfaction: the degree of gratification or pleasure of the player … highly subjective attribute

Learning: the facility to understand and "dominate" the game system …

Efficiency: the necessary time and resources to offer fun and entertainment to players while they achieve the different game objectives … an efficient video game is able to catch the player's attention from the first instant …

Immersion: the capacity to believe in the video game contents and integrate the player in the virtual game world…

Sánchez, J. G., Zea, N. P., & Gutiérrez, F. L. (2009, August). Playability: How to identify the player experience in a video game. In *IFIP Conference on Human-Computer Interaction* (pp. 356-359). Springer, Berlin, Heidelberg.

Gameplay Programming
What is… ?

Playability

Playability is defined as: a set of properties that describe the player experience using a specific game system… characterized by different attributes and properties to measure the video game player experience. *González Sánchez et al.* [19]

Motivation: the characteristics that provoke the player to realize concrete actions …
… to obtain a high degree of motivation, the game should have a set of resources to ensure the players perseverance in the performed actions to overcome the game challenges …

Emotion: the involuntary impulse, originated in response to the stimulus of the video game and induces feelings or unleashes automatic reactions and conducts….

Socialization: the degree of the set of game attributes, elements and resources that promote the social factor of the game experience in group…

Sánchez, J. G., Zea, N. P., & Gutiérrez, F. L. (2009, August). Playability: How to identify the player experience in a video game. In *IFIP Conference on Human-Computer Interaction* (pp. 356-359). Springer, Berlin, Heidelberg.

Gameplay Programming
What is… ?

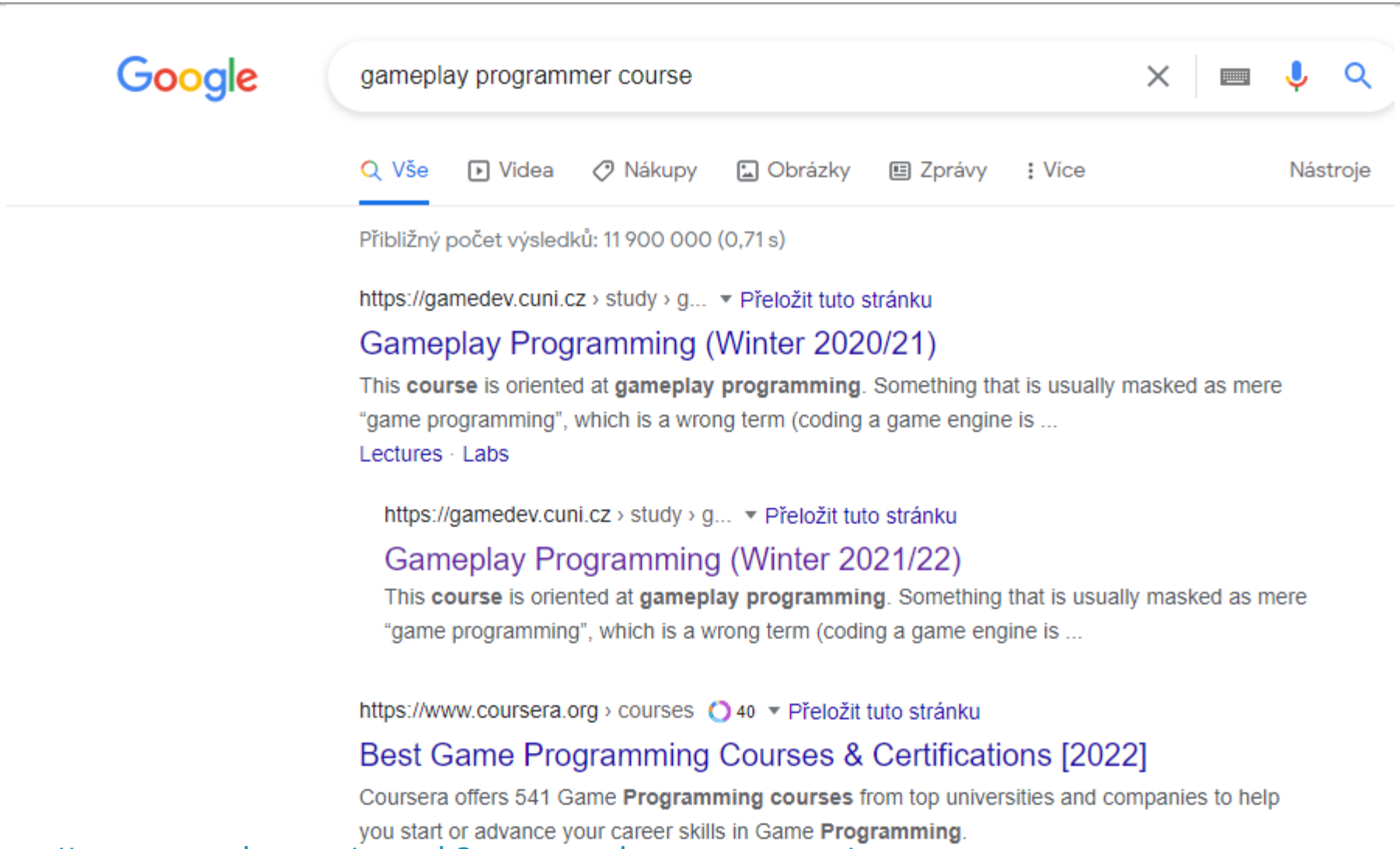
Gameplay Programmer
https://en.wikipedia.org/wiki/Video_game_programmer#Gameplay_programmer

Though all programmers add to the content and experience that a game provides, a gameplay programmer focuses more on a game's strategy, **implementation of the game's mechanics** and logic, and the **"feel" of a game**. This is usually not a separate discipline, as what this programmer does usually differs from game to game, and they will inevitably be involved with more specialized areas of the game's development such as graphics or sound.

⇒ Let's try to become a Gameplay Programmer!
https://www.google.com/search?q=gameplay+programming+course

Gameplay Programming

What is… ?



https://www.google.com/search?q=gameplay+programming+course

Gameplay Programming

What is… ?

SCOPUS

Gameplay Programming
What is… ?

SCOPUS

TITLE-ABS-KEY (gameplay AND programming AND syllabus)

0 hits

TITLE-ABS-KEY (gameplay AND programming AND curriculum)

10 hits, 0 relevant

TITLE-ABS-KEY (gameplay AND programming AND teach)

12 hits, 0 relevant

Gameplay Programming

What is… ?

Scholar Google

https://scholar.google.com/scholar?q=gameplay+programming

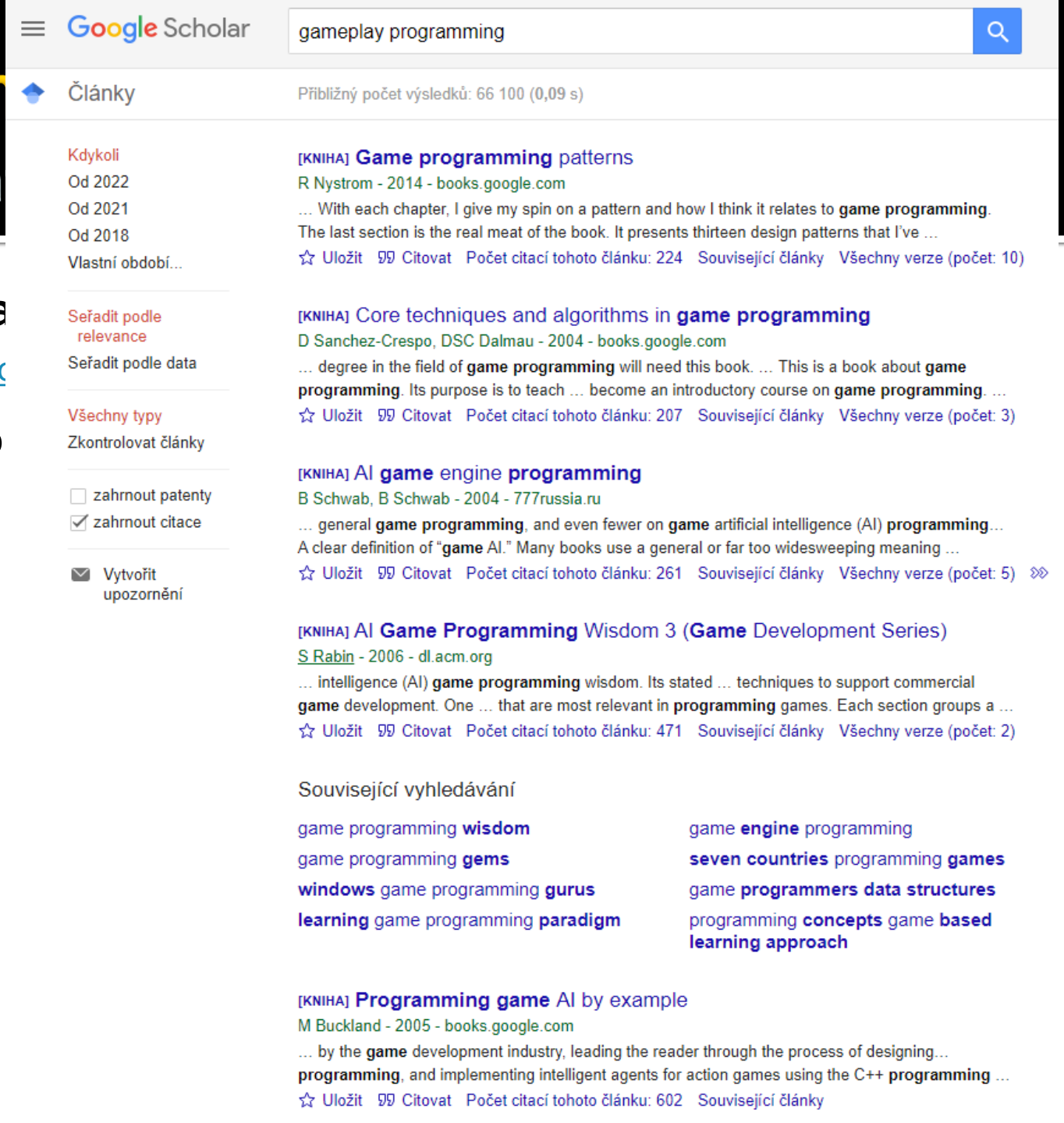
66 100 hits

Gar
Wha

Schola

https://sc

66 100



Google Scholar

gameplay programming

Články

Přibližný počet výsledků: 66 100 (0,09 s)

Kdykoli
Od 2022
Od 2021
Od 2018
Vlastní období...

Seřadit podle relevance
Seřadit podle data

Všechny typy
Zkontrolovat články

zahrnout patenty
zahrnout citace

Vytvořit upozornění

[KNIHA] **Game programming** patterns
R Nystrom - 2014 - books.google.com
... With each chapter, I give my spin on a pattern and how I think it relates to **game programming**. The last section is the real meat of the book. It presents thirteen design patterns that I've ...
☆ Uložit 99 Citovat Počet citací tohoto článku: 224 Související články Všechny verze (počet: 10)

[KNIHA] Core techniques and algorithms in **game programming**
D Sanchez-Crespo, DSC Dalmau - 2004 - books.google.com
... degree in the field of **game programming** will need this book. ... This is a book about **game programming**. Its purpose is to teach ... become an introductory course on **game programming**. ...
☆ Uložit 99 Citovat Počet citací tohoto článku: 207 Související články Všechny verze (počet: 3)

[KNIHA] AI **game** engine **programming**
B Schwab, B Schwab - 2004 - 777russia.ru
... general **game programming**, and even fewer on **game** artificial intelligence (AI) **programming**... A clear definition of "**game** AI." Many books use a general or far too widesweeping meaning ...
☆ Uložit 99 Citovat Počet citací tohoto článku: 261 Související články Všechny verze (počet: 5) »

[KNIHA] AI **Game Programming** Wisdom 3 (**Game** Development Series)
S Rabin - 2006 - dl.acm.org
... intelligence (AI) **game programming** wisdom. Its stated ... techniques to support commercial **game** development. One ... that are most relevant in **programming** games. Each section groups a ...
☆ Uložit 99 Citovat Počet citací tohoto článku: 471 Související články Všechny verze (počet: 2)

Související vyhledávání

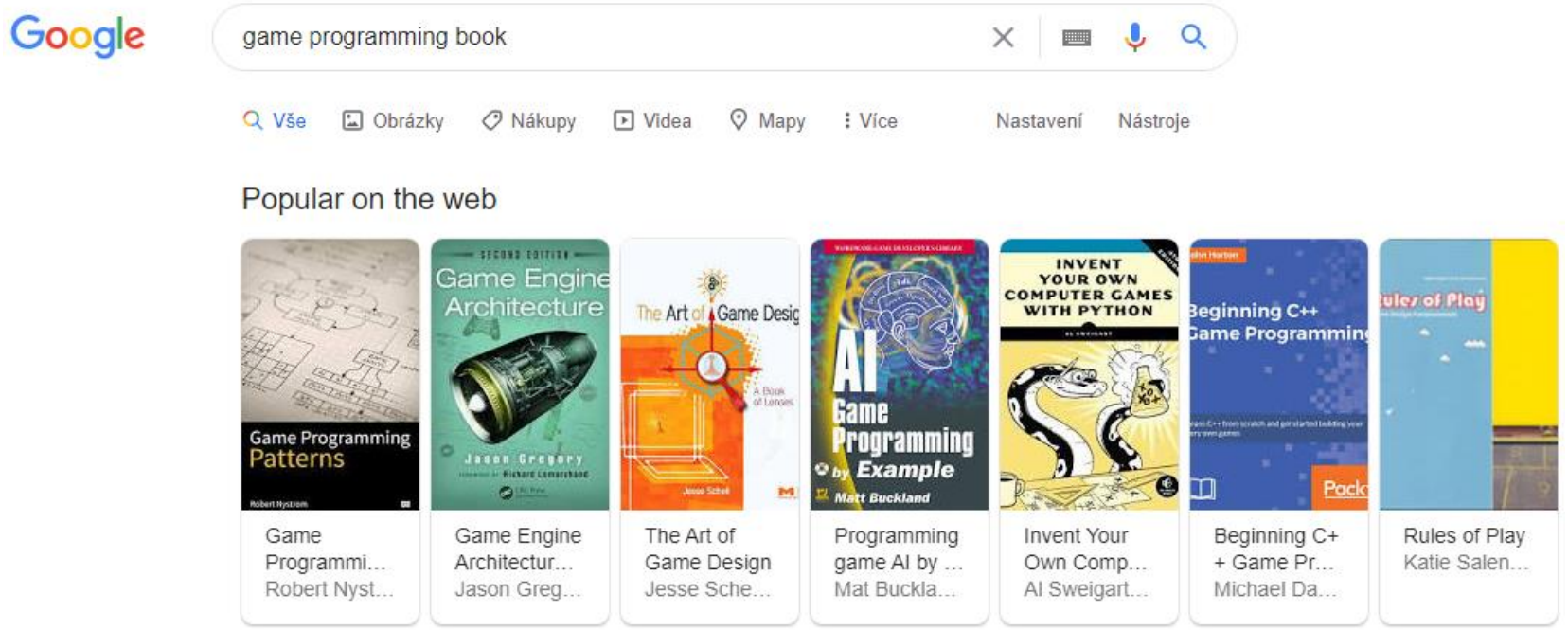
game programming **wisdom**
game programming **gems**
windows game programming **gurus**
learning game programming **paradigm**
game **engine** programming
seven countries programming **games**
game **programmers data structures**
programming **concepts** game **based learning approach**

[KNIHA] **Programming game** AI by example
M Buckland - 2005 - books.google.com
... by the **game** development industry, leading the reader through the process of designing... **programming**, and implementing intelligent agents for action games using the C++ **programming** ...
☆ Uložit 99 Citovat Počet citací tohoto článku: 602 Související články

Gameplay Programming
What is… ?

Another search

https://www.google.com/scholar?q=game+programming+book



Gameplay Programming

What is… ?

You know… we all can name a game with a memorable gameplay.


Portal
Do not look directly at the operational e
The Darkest Dungeon
MegaCrit the Ironclad 47/87 423 1 24th Floor
09-28-2017
Slay the Spire
End Turn
Swine Prince
HP: 132/132
DODGE: 0
SPD: 0
BEAST
Stun Resist: 100%
Blight Resist: 0%
Bleed Resist: 0%
Debuff Resist: 25%
Hero to Hit: 75%
Hero to Crit: 8%
Hero Dmg: 7 - 14
The Binding of Isaac
Stenley Parable

Gameplay Programming

What is… ?

You might object that those games are memorable because of their **GAME DESIGN**.

But who does bring a GAME DESIGN and its GAME MECHANICS into LIFE?

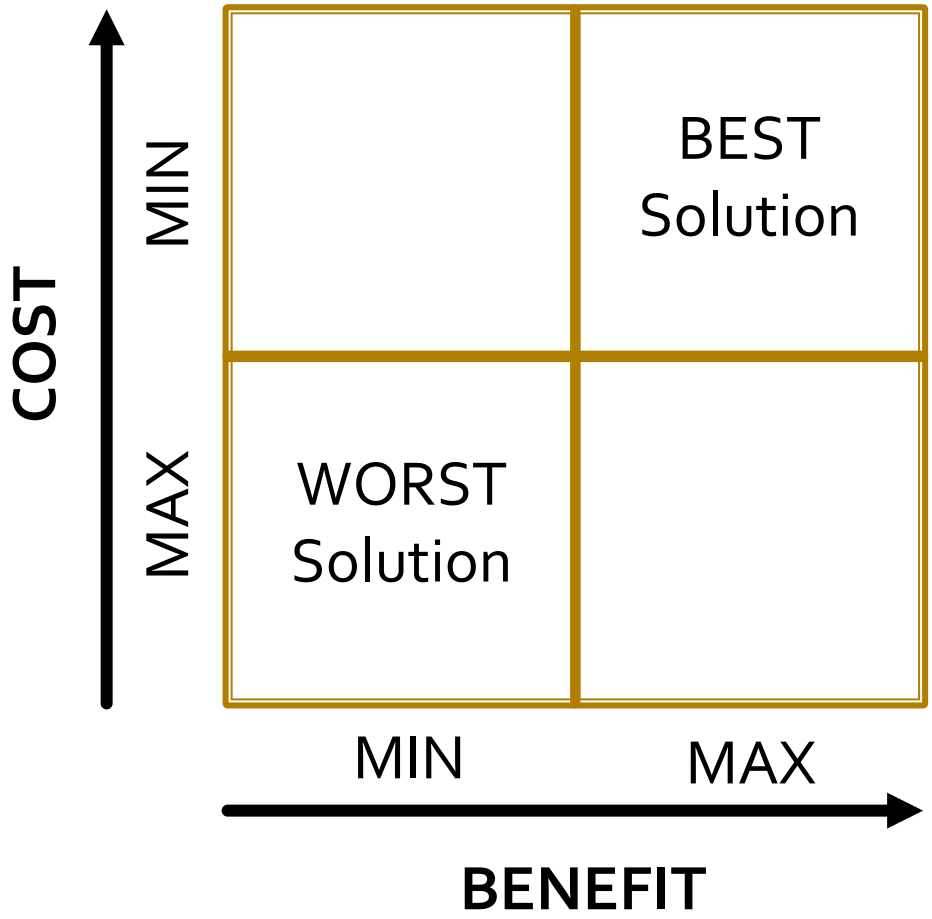
Bingo ☺

a GAMEPLAY PROGRAMMER!

Gameplay Programming

What is… ?

Sanchez-Crespo, D., & Dalmau, D. S. C. (2004). *Core techniques and algorithms in game programming*. New Riders.



Gameplay Programming

What this course is going to be really about?

Bits of GPU pipeline

Game programming patterns

Scene tree / hierarchy / "prefabs"

Tweening, paths, curves, time

2D Textures, spritesheets, spritesheet animations, optimizations

3D models and animations (peek)

Input handling, character controllers (peek)

~~UI, Dialogue systems, Localization~~

Physics (peek), colliders, joints, triggers, callbacks

Music and sounds

Random number generators

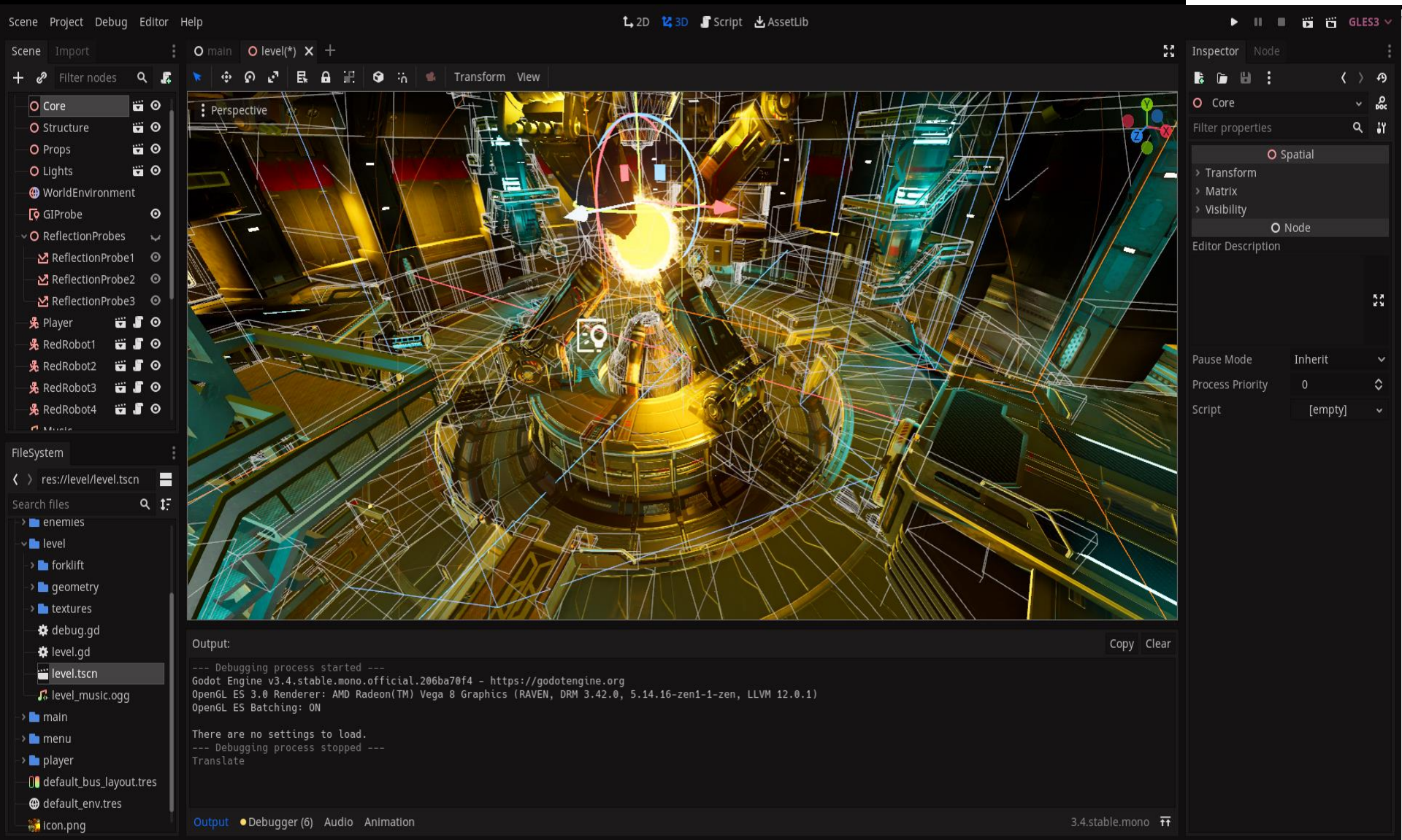
~~Serialization~~

~~+~~

Godot => GDScript (a different Python of a sort)

GODOT

Showtime



Homework

Describe a game that (you can say) has a memorable gameplay

1. Play your favorite game
2. Explore your emotions and describe what you like about the gameplay (write it down to some online doc); use the prism of playability, try to see the gameplay from different angles. => check the next slide for structure!
3. Similarly to what I will show you with Slay the Spire
   a. Play a game again
   b. Get to a place where your favourite gameplay is clearly visible and shoot the game in action (~20-30 secs is sufficient!)
   c. Put the recording online somewhere (YouTube unlisted video perhaps?) and save the link to the video into the document
   d. Describe as technically as possible (through the lens of gameplay programming) what is happening on the screen.
4. Optionally share your document with the group via Discord channel #msc-ncgd003-gameplay-programming
5. Sent the report to us (e.g., via email or DM at Discord)

Discord invite link: https://discord.gg/c49DHBJ

Homework

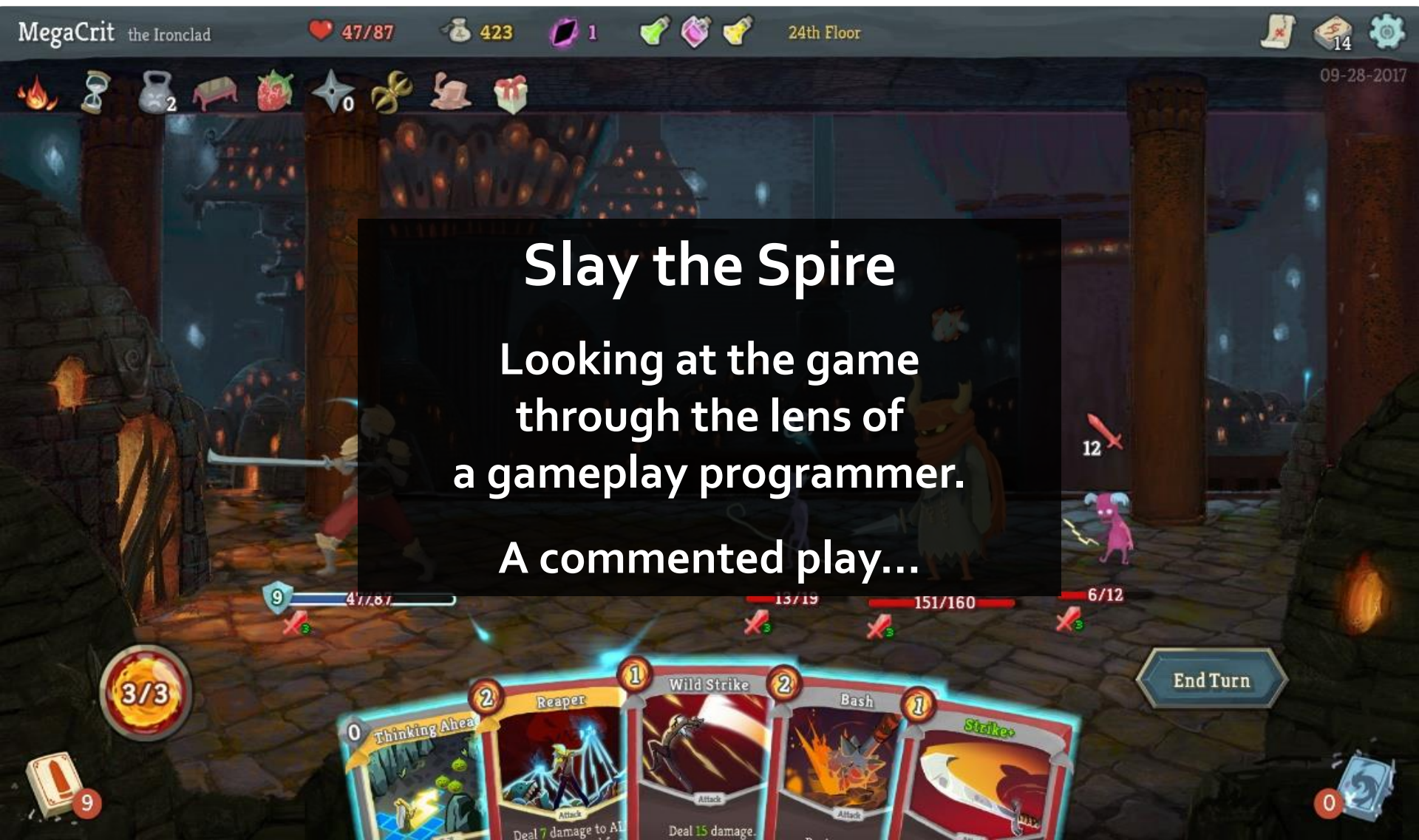
Describe a game that (you can say) has a memorable gameplay

Document structure

- Describe the game in your own words
- Share your thoughts on why the game is great from your point of view, what do you enjoy?
- Go through the list of playability properties and describe the game through these lens
  1. Satisfaction - What do you find satisfying in the game? What not? State also your reasons.
  2. Learning - How is the game facilitating the learning? Could you draw your learning curve (x-axis is time, y-axis is mastery, not to be confused with challenge!)
  3. Efficiency - How the game is getting your attention? Are there any weak spots during play sessions?
  4. Immersion - What elements within the game are contributing to the player's immersion? Is there anything, which is preventing it or could be made better?
  5. Motivation - How challenging the game is? Could you draw a graph (x-axis is time, y-axis is challenge) of your experience playing the game (could be a single session).
  6. Emotion - What emotions the game draws out of you? Do you enjoy them? Do they contribute to immersion/motivation/etc.?
  7. Socialization - Beside this course, have you talked about this game to anyone (IRL or online) or have you been reading other player's reports/experiences with the game? What were you talking about or what were you interested in to hear or read?
- Link to the commented video

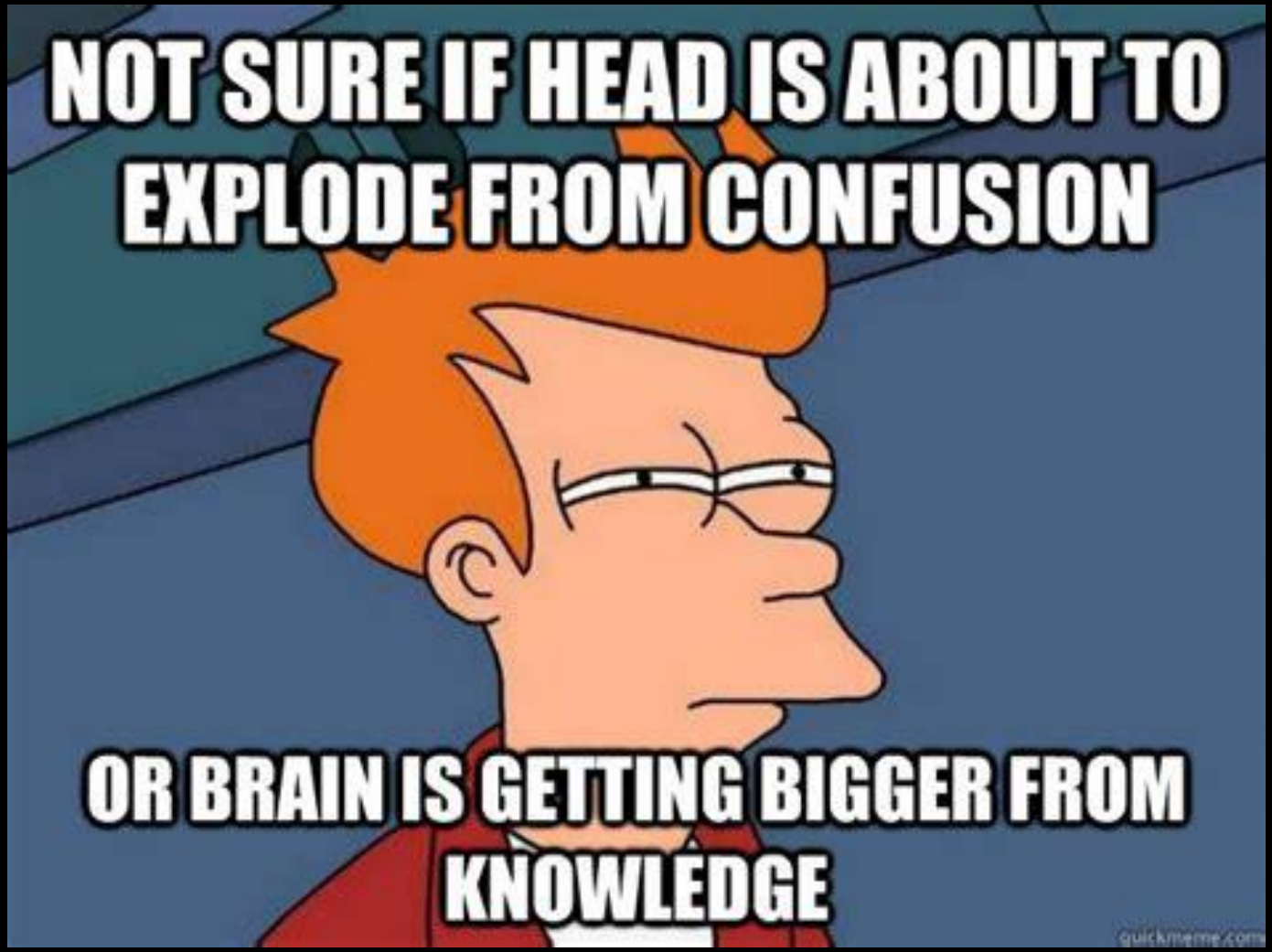
Gameplay Programming

What is… ?



Gameplay Programming

What is… ?



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Material has been produced within and supported by the project „Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“ kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.

Faculty of Mathematics and Physics
Charles University, Prague
10th October 2024

GAME DEVELOPMENT @CUNI.CZ

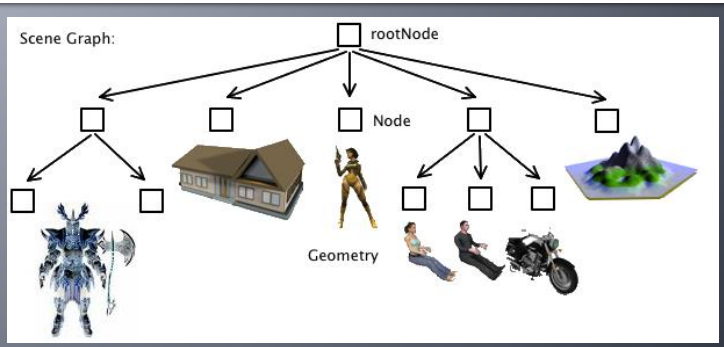
matfyz

Gameplay Programming Level 02

Objects/Entities/Systems

Modeling your game scene

In an hour or so…



The Game Loop

Where it all begins

FUNDAMENTALS
OF
GAME ARCHITECTURE

The Game Loop
Where it all begins

We said that … game play programming is about trying to manipulate that interface between the player and the game.

So we need a game to start with…

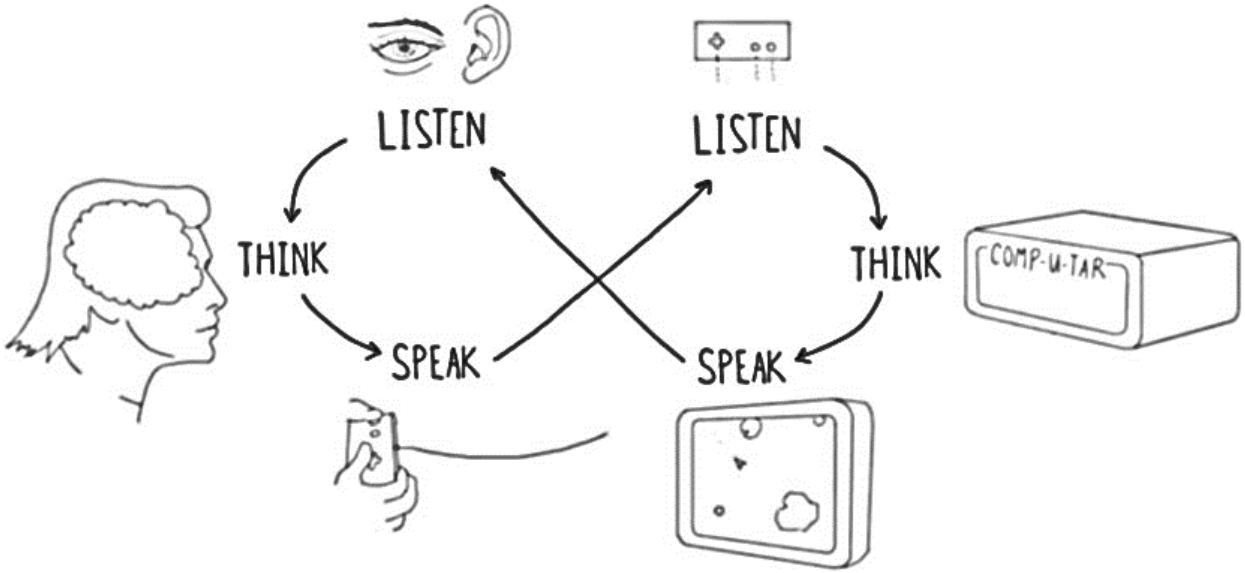


Image: Swink, S. (2008). Game feel: a game designer's guide to virtual sensation. CRC Press.

The Game Loop
Where it all begins

We said that … game play programming is about trying to manipulate that interface between the player and the game.

So we need a game to start with…

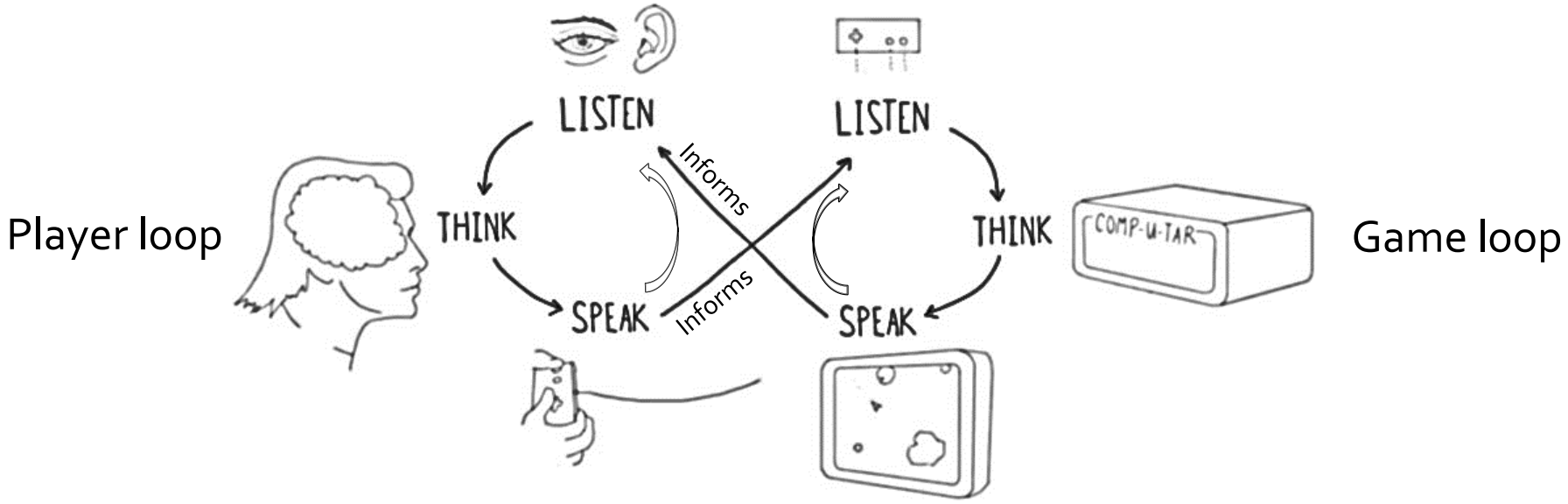


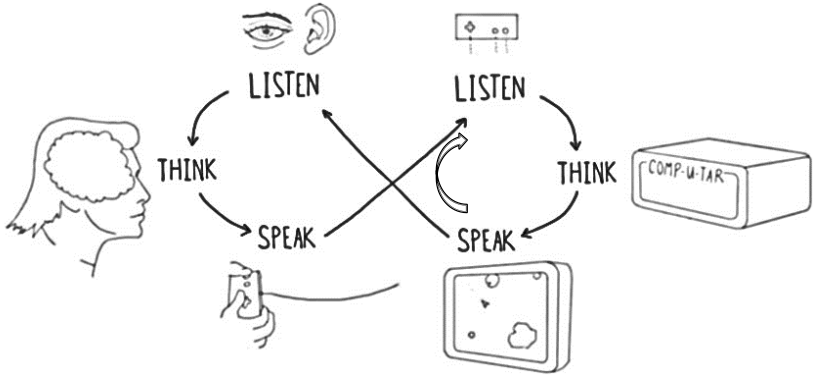
Image: Swink, S. (2008). Game feel: a game designer's guide to virtual sensation. CRC Press.

The Game Loop
Where it all begins

We said that … game play programming is about trying to manipulate that interface between the player and the game.

So we need a game to start with…

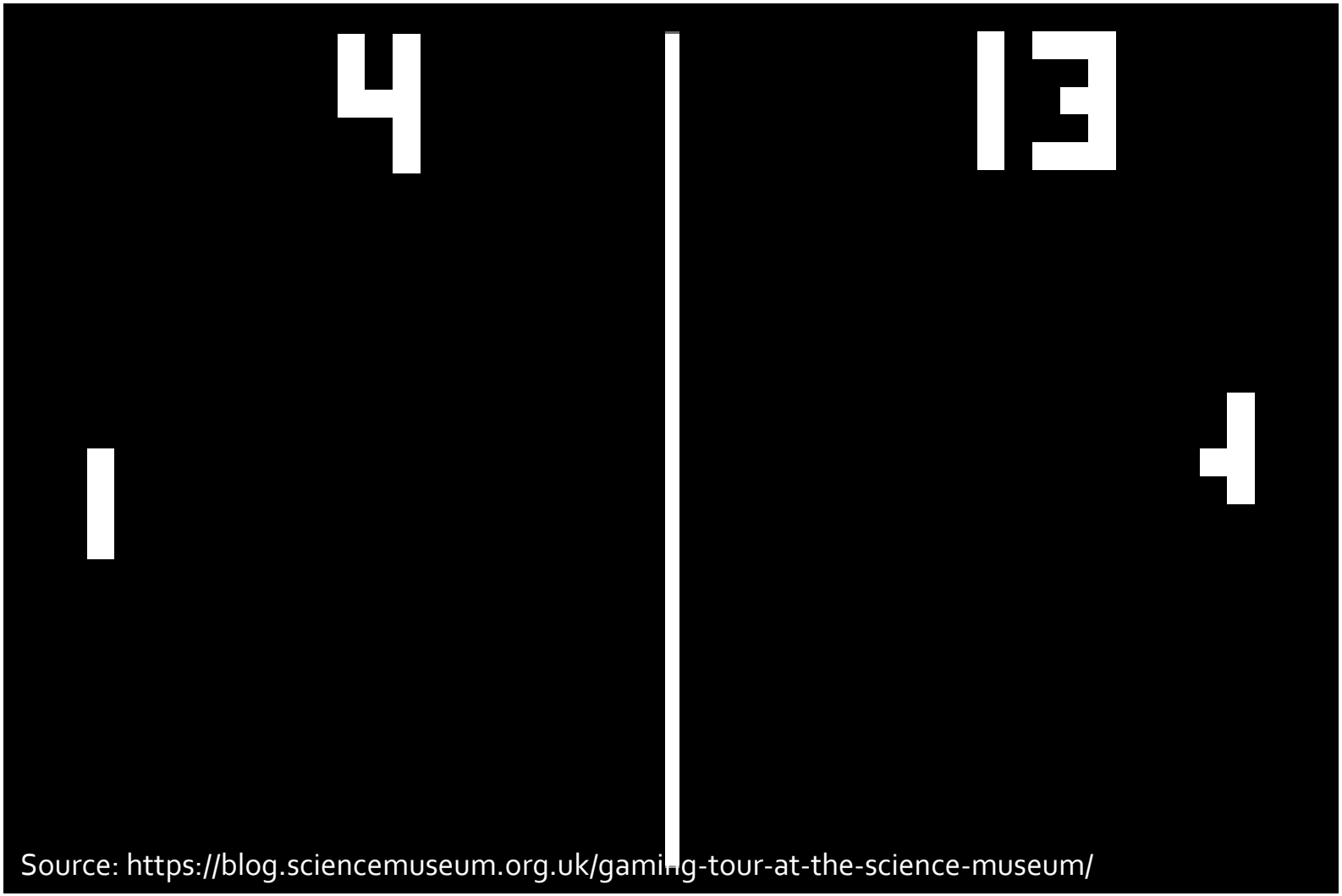
```
void main() {
    setupGameWorld();
    while (true) {
        readPlayerInput();
        if (quitRequested()) break;
        updateGameWorld();
        renderGameWorld();
        produceAudio();
    }
    // don't clean up, OS will do ☺
}
```



Game loop model: Single-threaded

The Game Loop

Where it all begins



Source: https://blog.sciencemuseum.org.uk/gaming-tour-at-the-science-museum/

```
startover:
SCREEN 10
CLS
PRINT "What is the name of player 1?"
INPUT player1.name$
PRINT "What is the name of player 2?"
INPUT player2.name$
PRINT "How many rounds do you want to play?"
INPUT round.total
player1.score = 0
player2.score = 0
rounds = 0
new:
rounds = rounds + 1
```

1. Reading initial input

```
'draw the field
SCREEN 7
CLS
PAINT (5, 5), (2)
LINE (0, 0)-(3, 199), 0, BF
LINE (0, 199)-(319, 194), 0, BF
LINE (318, 199)-(314, 0), 0, BF
PAINT (1, 1), (15), (2)
LINE (0, 0)-(0, 200), 0
LINE (1, 74)-(3, 144), 0, BF
LINE (314, 74)-(319, 144), 0, BF
LINE (1, 0)-(319, 20), 15, BF
LINE (0, 0)-(320, 15), 0, BF
LINE (317, 0)-(319, 199), 0, BF
LOCATE 1, 1
```

3. Rendering

```
IF player1.score >
player2.score THEN
COLOR 9
ELSE
COLOR 12
END IF
IF player1.score =
player2.score THEN
COLOR 15
END IF
PRINT "Score:"; "   ";
player2.name$; ":";
player2.score; "   ";
player1.name$; ":";
player1.score

IF rounds > round.total
THEN
GOTO quit
END IF

player1.x = 300
player1.y = 95
player2.x = 12
player2.y = 95
player1.move$ = "n"
player2.move$ = "n"
ball.x = 163
ball.y = 3 * INT(55 * RND)
+ 26
```

4. Quit check

```
x = INT(2 * RND)
IF x = 1 THEN
ball.ns = 3
ELSE
ball.ns = -3
END IF

x = INT(2 * RND)
IF x = 1 THEN
ball.ew = 3
ELSE
ball.ew = -3
END IF

ball.x.2 = ball.x
ball.y.2 = ball.y
player1.x.2 = player1.x
player1.y.2 = player1.y
player2.x.2 = player2.x
player2.y.2 = player2.y
player1.ns = 0
player1.ew = 0
player2.ns = 0
player2.ew = 0

a:
CIRCLE (ball.x.2, ball.y.2), 6, 2
PAINT (ball.x.2, ball.y.2), (2), (2)

LINE (player1.x.2, player1.y.2 + 1)-
(player1.x.2 + 6, player1.y.2 + 23), 2, BF
LINE (player2.x.2, player2.y.2 + 1)-
(player2.x.2 + 6, player2.y.2 + 23), 2, BF
LINE (player1.x, player1.y + 1)-(player1.x + 6,
player1.y + 23), 9, BF
LINE (player2.x, player2.y + 1)-(player2.x + 6,
player2.y + 23), 12, BF
CIRCLE (ball.x, ball.y), 6, 14
PAINT (ball.x, ball.y), (14), (14)

FOR count = 1 TO Flicker.Control
NEXT count

ball.x.2 = ball.x
ball.y.2 = ball.y
player1.x.2 = player1.x
player1.y.2 = player1.y
player2.x.2 = player2.x
player2.y.2 = player2.y

ball.x = ball.x + ball.ew
ball.y = ball.y + ball.ns
player1.x = player1.x + player1.ew
player1.y = player1.y + player1.ns
player2.x = player2.x + player2.ew
```

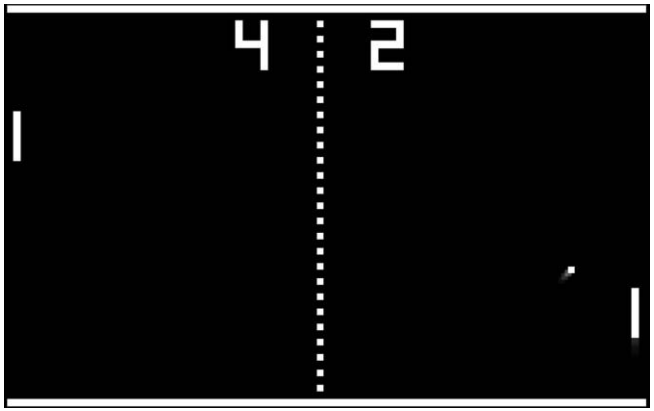
2. Definitions of game objects

```
IF ball.x - 6 <= 6 AND ball.y >= 74 AND ball.y <= 144 THEN
GOTO goal.1
END IF

IF ball.x + 6 >= 312 AND ball.y >= 74 AND ball.y <= 144 THEN
GOTO goal.2
END IF

IF ball.y - 6 <= 20 THEN
ball.ns = -3
END IF
IF ball.y + 6 >= 194 THEN
ball.ns = -3
END IF

IF ball.x - 6 <= 6 THEN
ball.ew = 3
END IF

IF ball.x + 6 >= 312 THEN
ball.ew = -3
END IF

IF ball.x + 6 >= player1.x AND ball.x - 6 <= player1.x + 6 AND ball.y
<= player1.y AND ball.y <= player1.y + 24 THEN
ball.ew = -ball.ew
IF noise = 1 THEN
BEEP
END IF
END IF

IF ball.x + 6 >= player2.x AND ball.x - 6 <= player2.x + 6 AND ball.y
>= player2.y AND ball.y <= player2.y + 24 THEN
ball.ew = -ball.ew
IF noise = 1 THEN
BEEP
END IF
END IF

IF player1.y <= 20 THEN
player1.ns = 3
END IF

IF player1.y + 24 >= 194 THEN
player1.ns = -3
END IF

IF player1.x <= 6 THEN
player1.ew = 3
END IF

IF player1.x + 6 >= 312 THEN
player1.ew = -3
END IF

IF player2.y <= 20 THEN
player2.ns = 3
END IF

IF player2.y + 24 >= 194 THEN
player2.ns = -3
END IF

IF player2.x <= 6 THEN
player2.ew = 3
END IF

IF player2.x + 6 >= 312 THEN
player2.ew = -3
END IF

moves = INKEY$
SELECT CASE moves

CASE "p"
LOCATE 12, 3
COLOR 15
PRINT "Game Paused, Hit Space to Continue"
WHILE INKEY$ <> ""
WEND
LINE (10, 44)-(310, 144), 2, BF

CASE "q"
GOTO quit

'player1 change in movement
CASE "8"
player1.ns = -3

CASE "6"
player1.ew = 3

CASE "2"
player1.ns = 3

CASE "4"
player1.ew = -3

CASE "5"
player1.ew = 0
player1.ns = 0

'player2 change in movement
CASE "i"
player2.ns = -3

CASE "l"
player2.ew = 3

CASE ","
player2.ns = 3

CASE "m"
player2.ns = 3

CASE "j"
player2.ew = -3

CASE "k"
player2.ew = 0
player2.ns = 0

END SELECT

GOTO a

goal.1:
LOCATE 14, 2
COLOR 9
PRINT "Goal!"
player1.score = player1.score + 1
WHILE INKEY$ <> ""
WEND
GOTO new

goal.2:
COLOR 4
LOCATE 14, 35
PRINT "Goal!"
player2.score = player2.score + 1
WHILE INKEY$ <> ""
WEND
GOTO new

quit:
LOCATE 12, 6
COLOR 4
PRINT "      Game Over      "
LOCATE 13, 6
PRINT "Do you want to play again? y/n"
DO
y.ns = INKEY$
LOOP UNTIL y.ns = "y" OR y.ns = "n"
IF y.ns = "y" THEN
GOTO startover
END IF
```

Source: https://jefflewis.net/programming-qbasic-pong20.html

The Game Loop

Where it all begins




```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

Update game world

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

How it is now

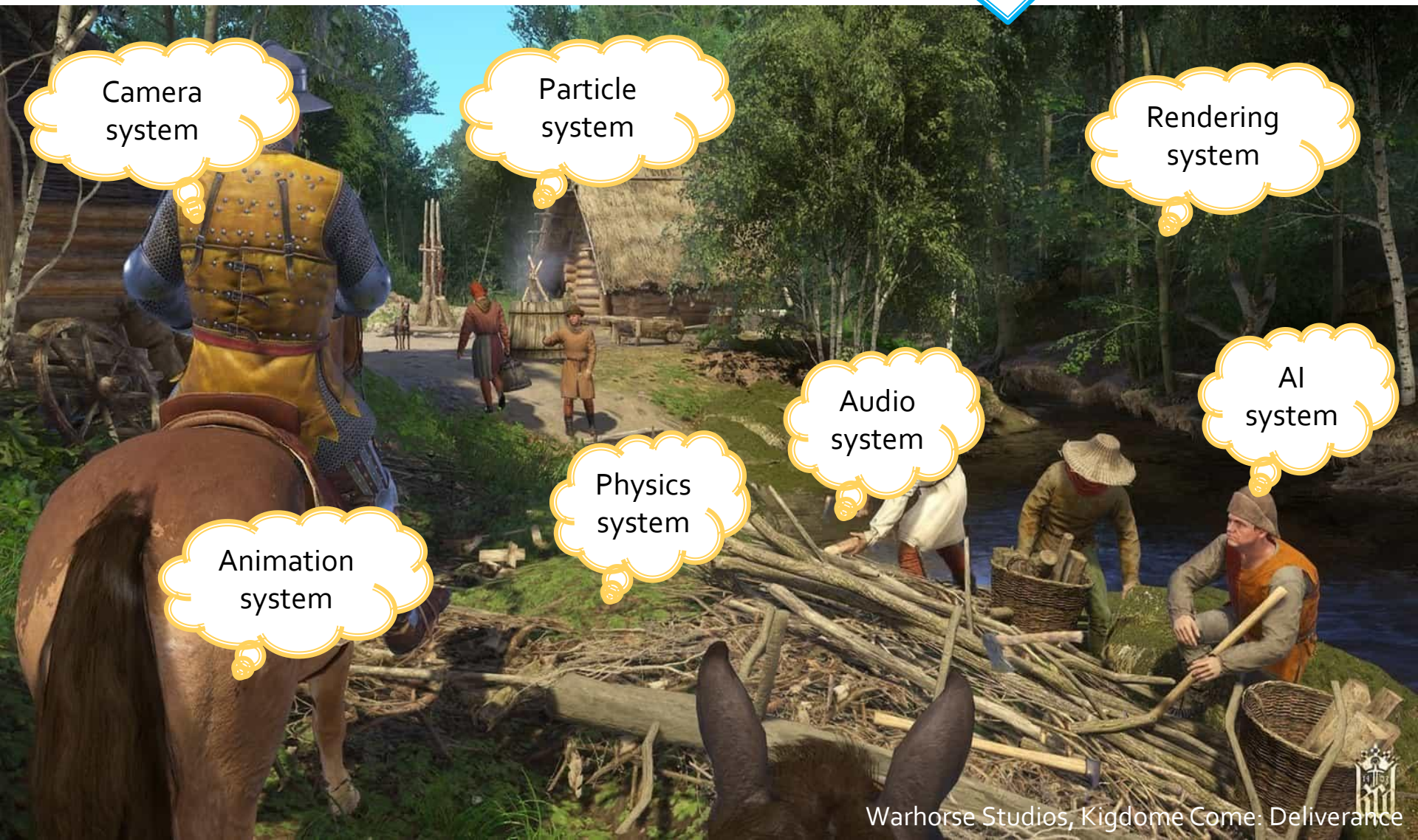
Number of objects would be much higher for a game like KC:D ...

Warhorse Studios, Kigdome Come: Deliverance

The Game Loop

How it is now

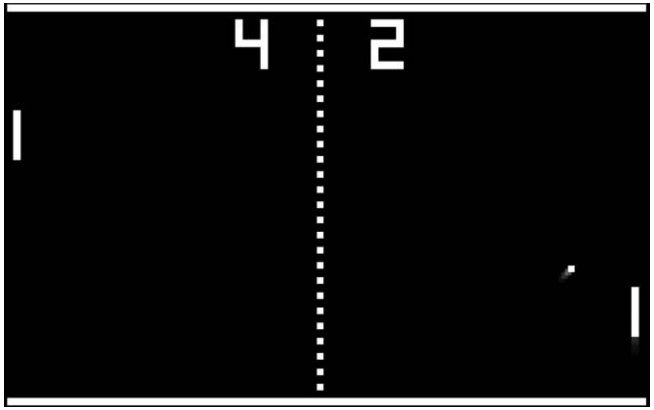
Abstracted code manipulating those objects would be complex even more…



Warhorse Studios, Kigdome Come: Deliverance

The Game Loop

Yet we can see the Pong through the "system" prism



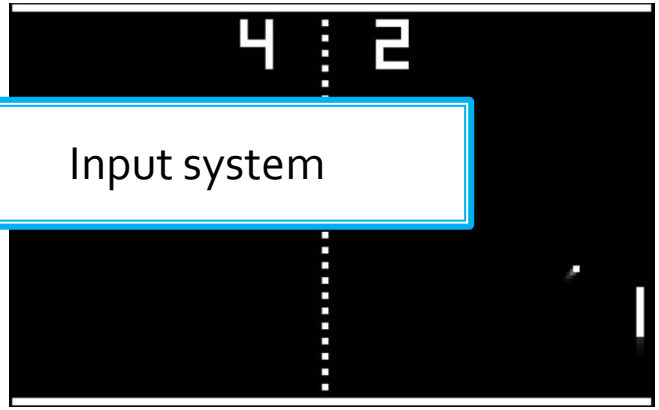

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

Yet we can see the Pong through the "system" prism

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incrementScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

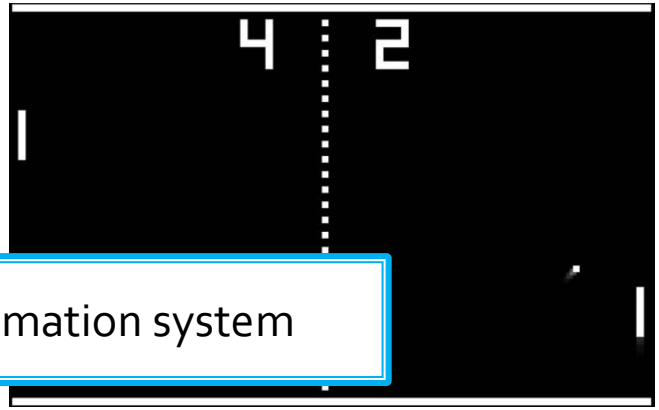


Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

Yet we can see the Pong through the "system" prism

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incrementScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

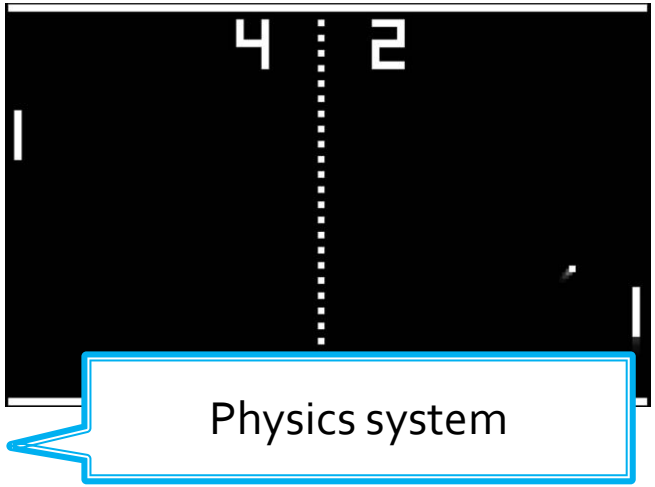


Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

Yet we can see the Pong through the "system" prism

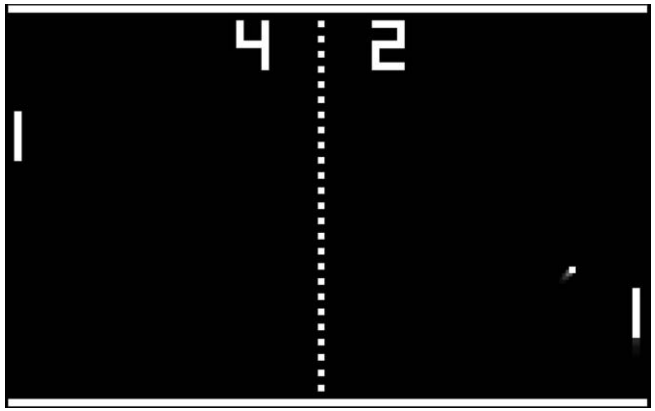
```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```



Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

Yet we can see the Pong through the "system" prism




```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

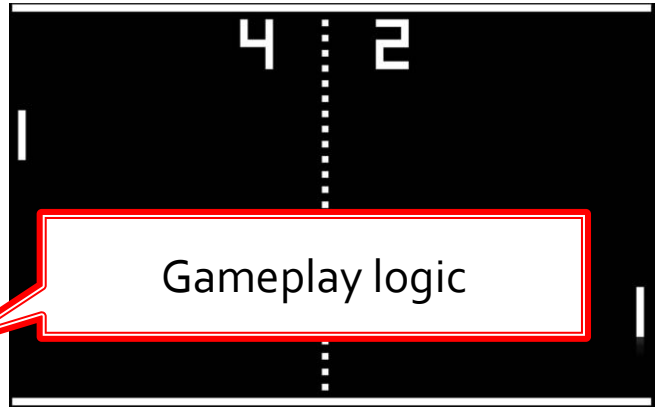
Rendering system

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Loop

Yet we can see the Pong through the "system" prism

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incrementScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```



Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

How it came to be

ABSTRACTING GAME OBJECTS

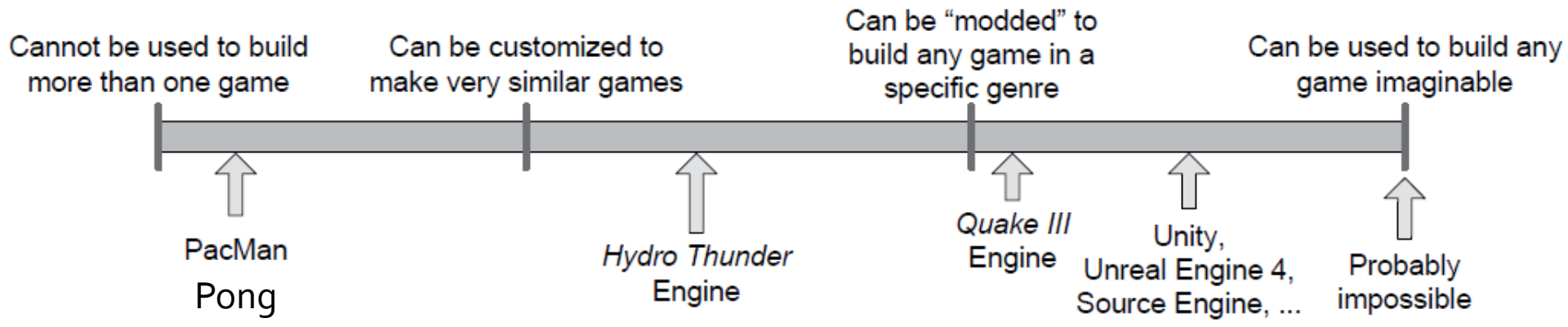
… why?

The Game Model

How it came to be

ABSTRACTING GAME OBJECTS

… why?



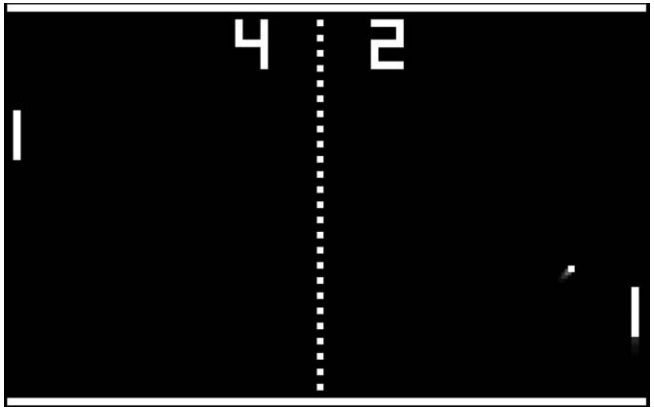
"Ad-hoc code" Pong

Pong reusing engine's code

Image: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

Pong as an engine?



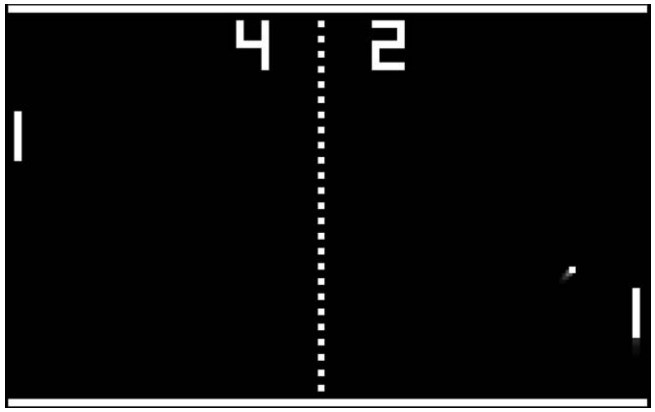

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

If we see this code as an architecture, we need to conclude not much else can be built with this architecture if we do not edit the code directly.

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

Expressing Pong code "an engine way"




```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAY
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```

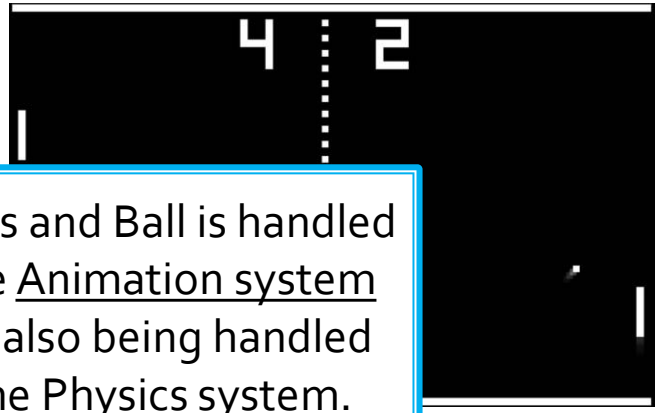
We need to abstract game objects in a way, they could be manipulated by different systems at once.

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

Expressing Pong code "engine way"

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```



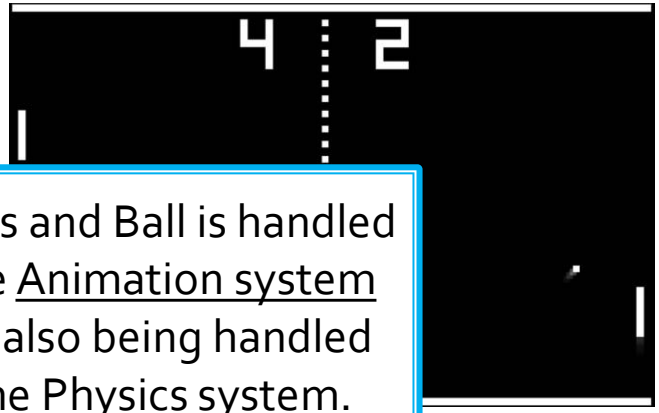
Paddles and Ball is handled by the Animation system while also being handled by the Physics system.

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

Expressing Pong code "engine way"

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT
            incrementScore(LEFT_PLAYER)
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```



Paddles and Ball is handled by the Animation system while also being handled by the Physics system.

How to model game objects so they can be easily defined, ideally using declarations only, while being suitable for digestions by different systems making today's game engines?

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

The Game Model

Disclaimer...

The Game Model
OOP Approach

“Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes.”

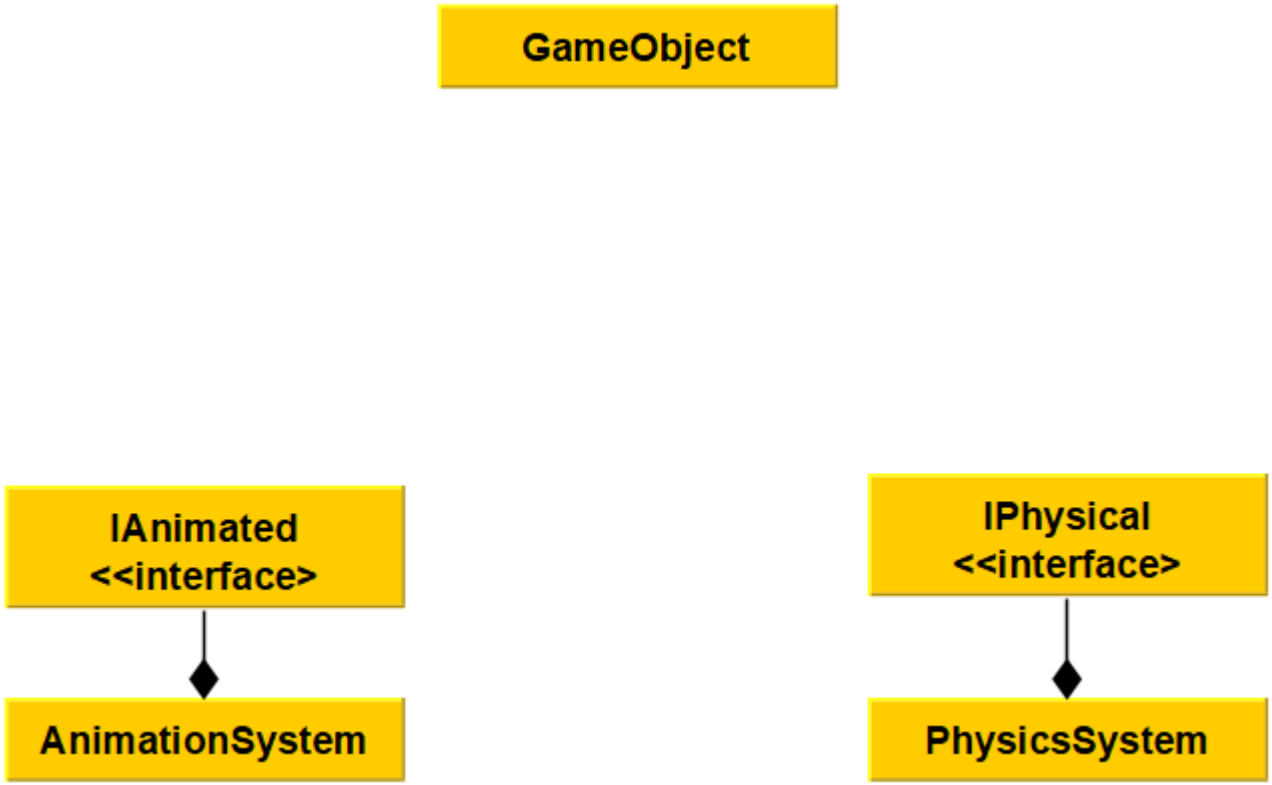
GameObject

AnimationSystem

PhysicsSystem

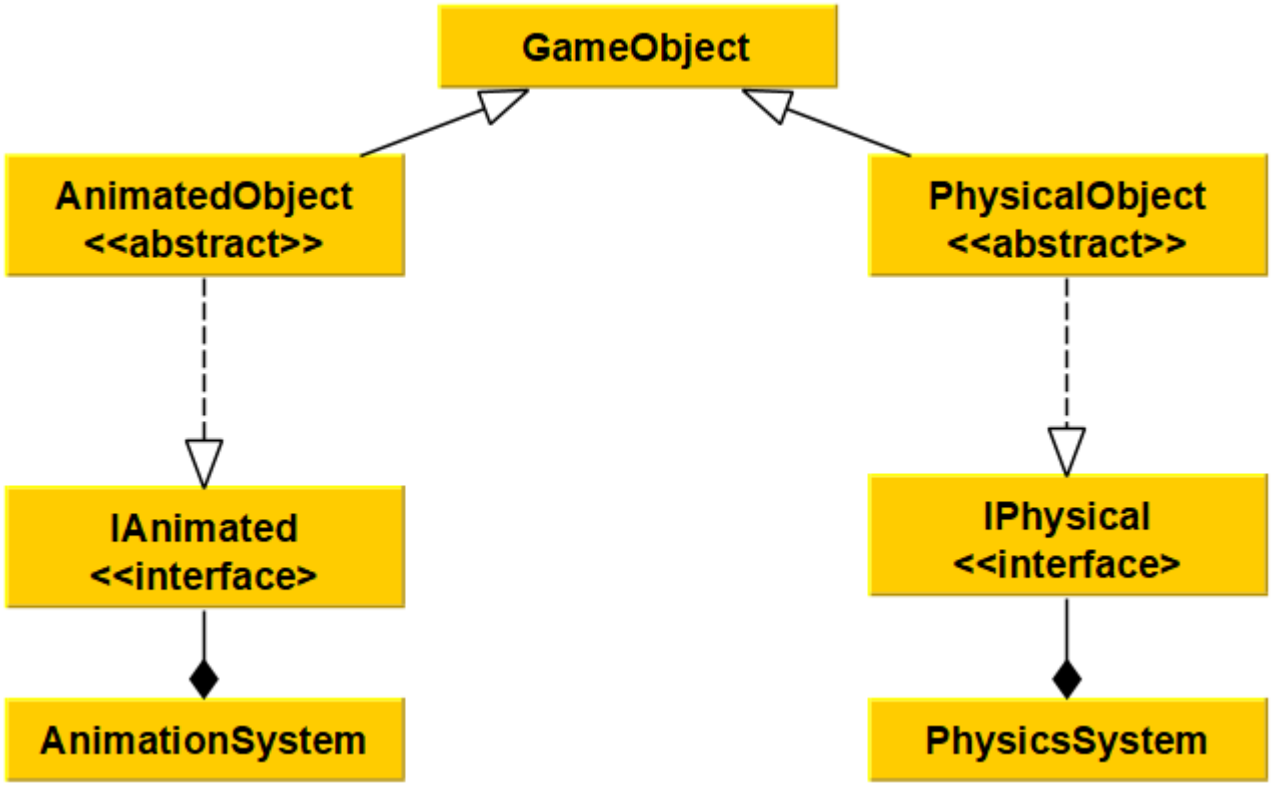
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



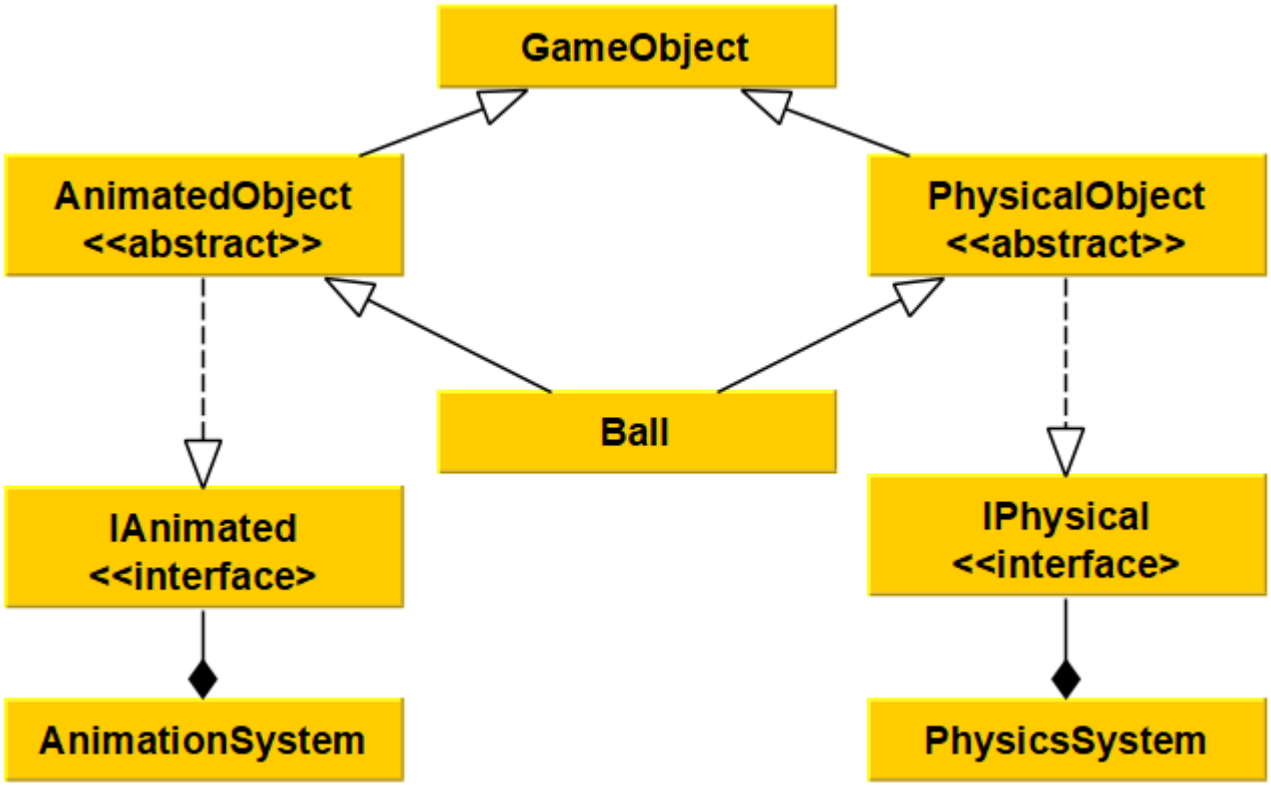
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



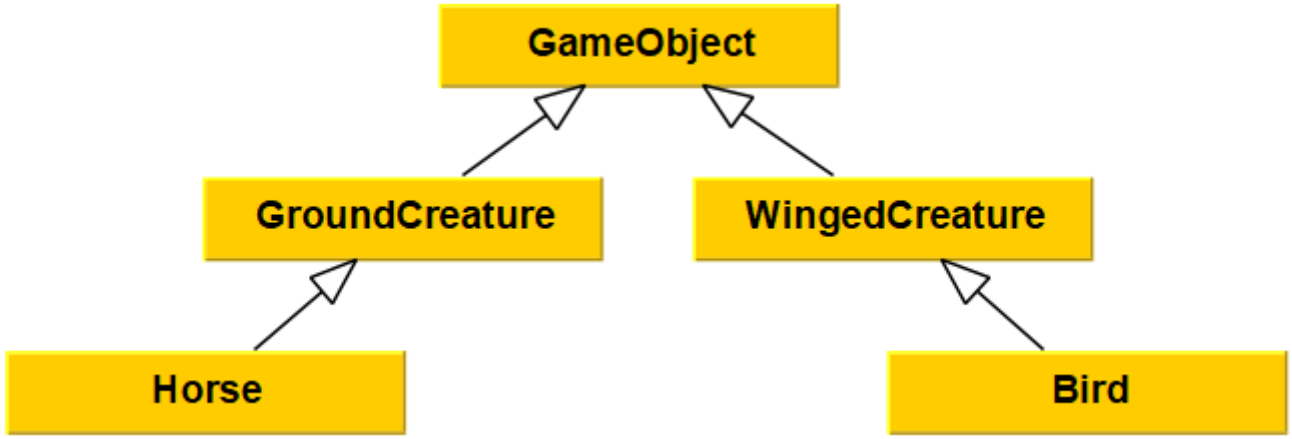
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



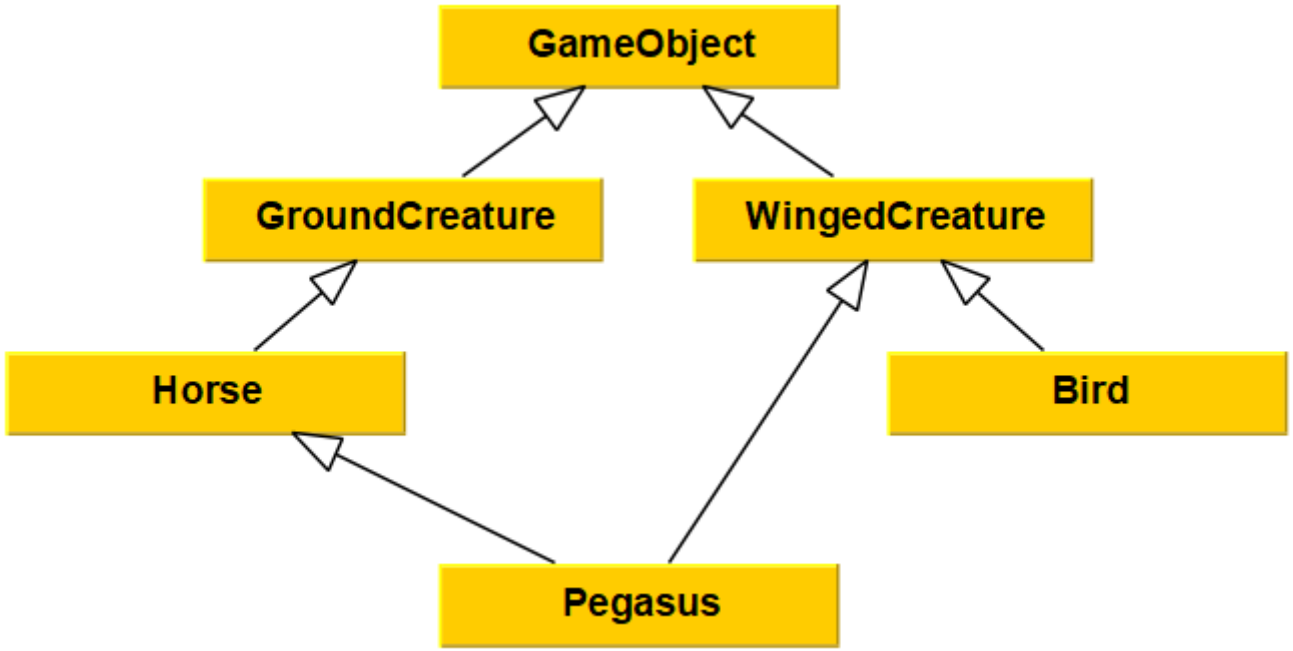
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



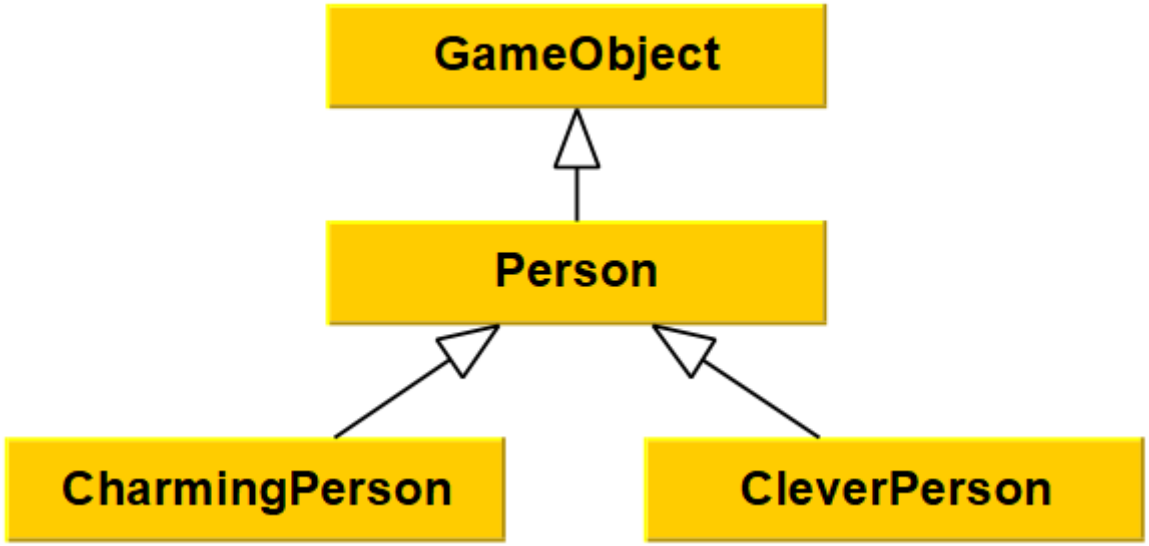
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



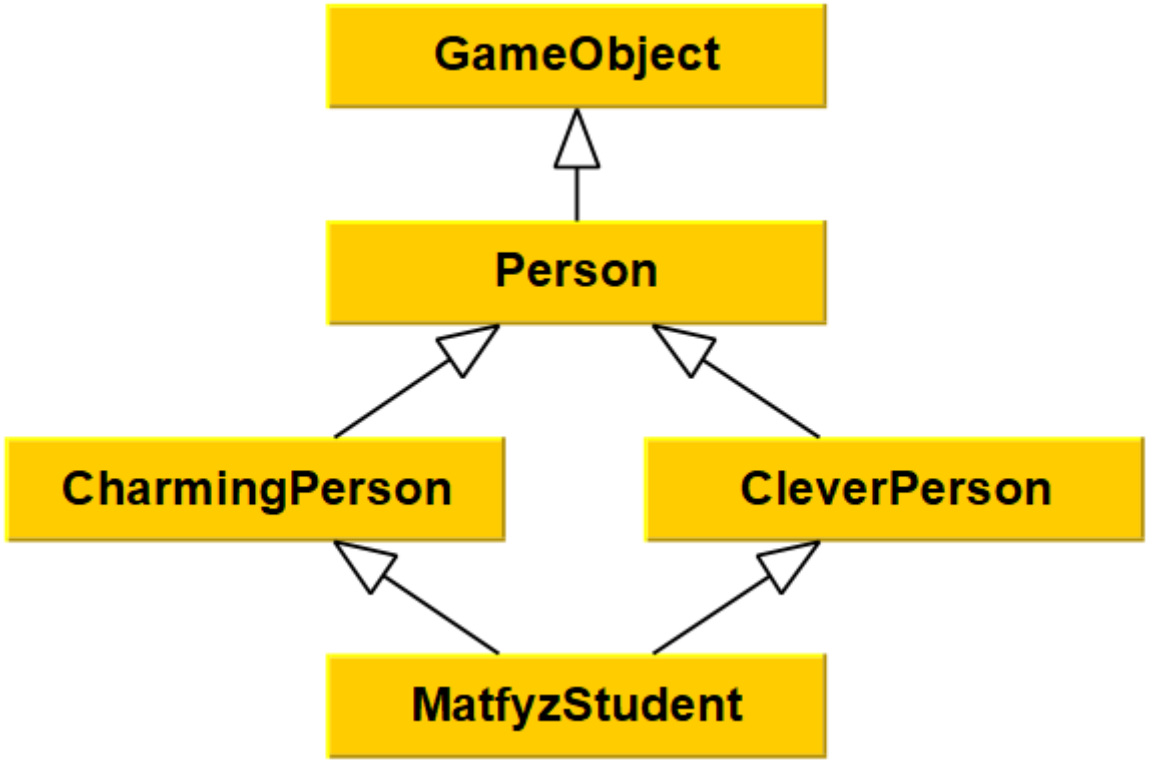
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



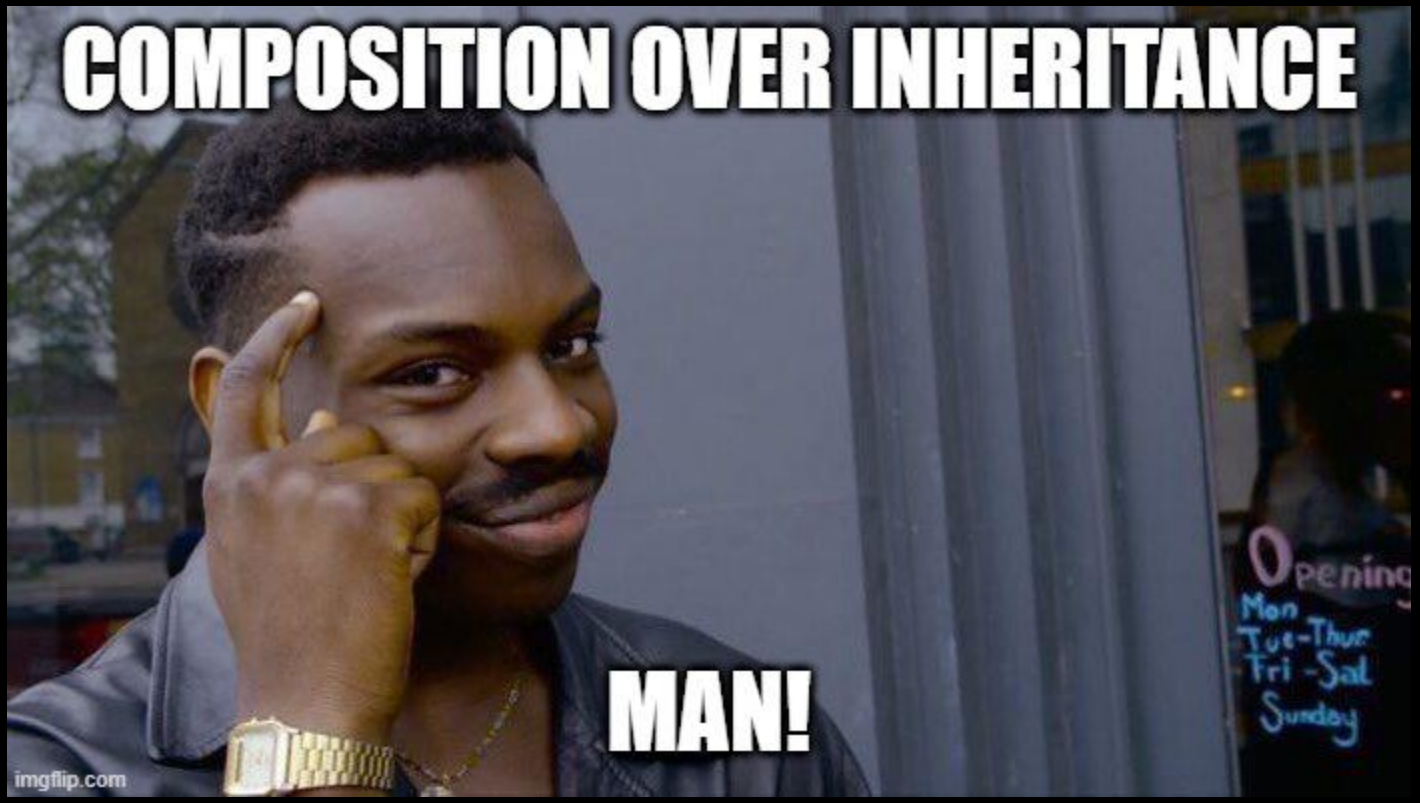
The Game Model
OOP Approach

"Languages that support object-oriented programming (OOP) typically use inheritance for code reuse and extensibility in the form of either classes or prototypes."



The Game Model

OOP Approach



The Ga de
C pproach

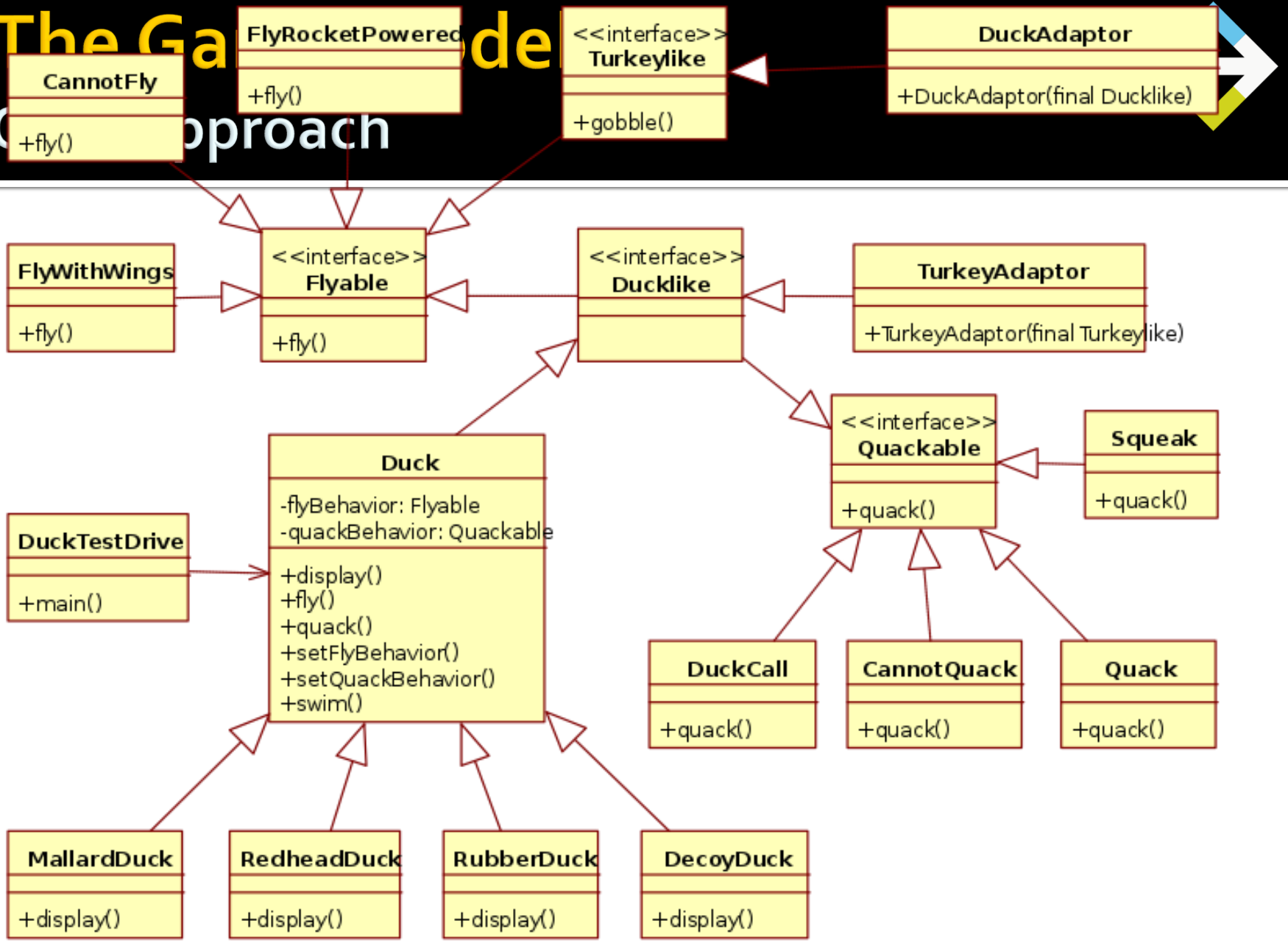


Image: https://en.wikipedia.org/wiki/File:UML_diagram_of_composition_over_inheritance.svg

The Ga de
C pproach

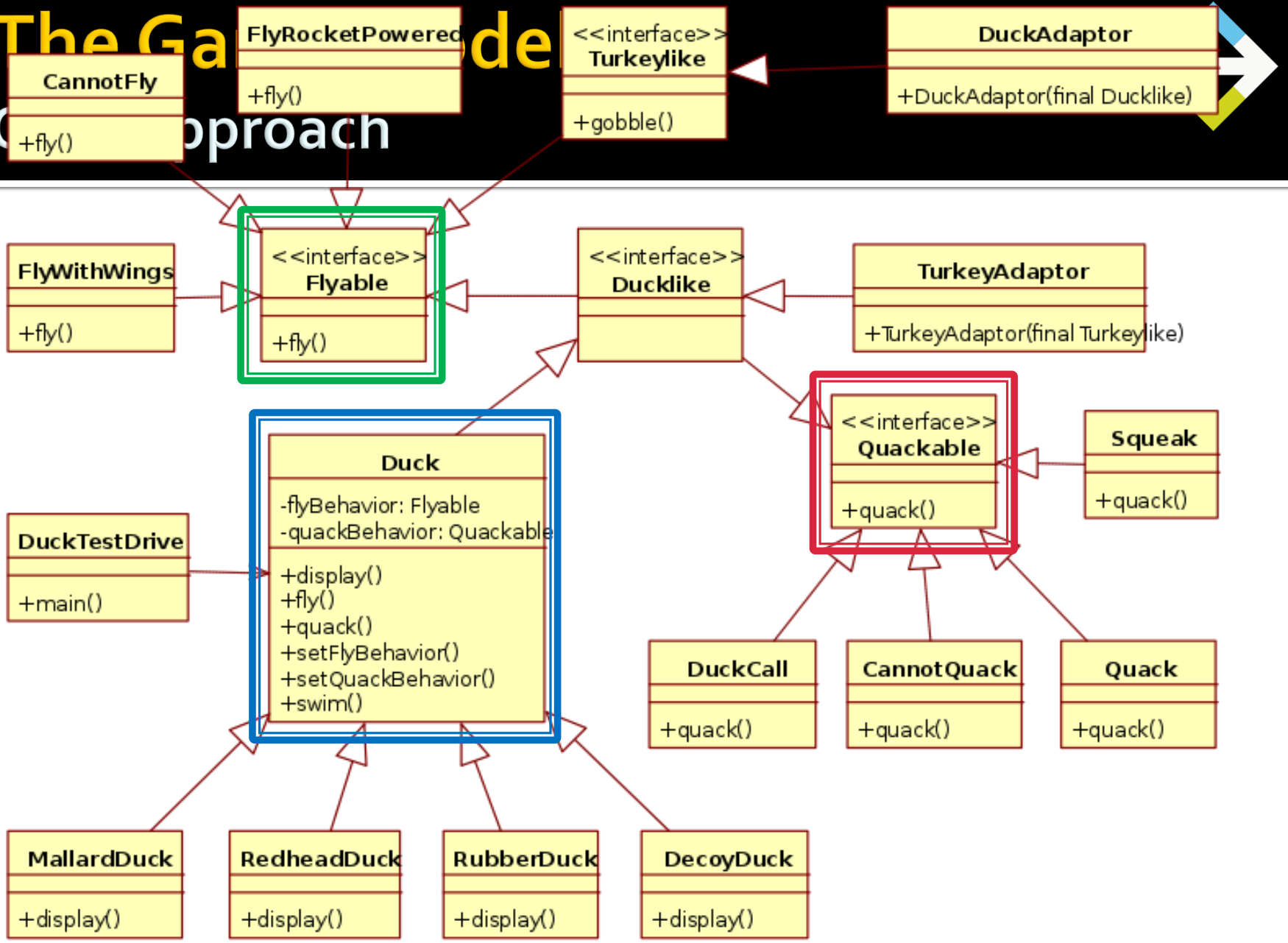


Image: https://en.wikipedia.org/wiki/File:UML_diagram_of_composition_over_inheritance.svg

The Ca de
( )proach

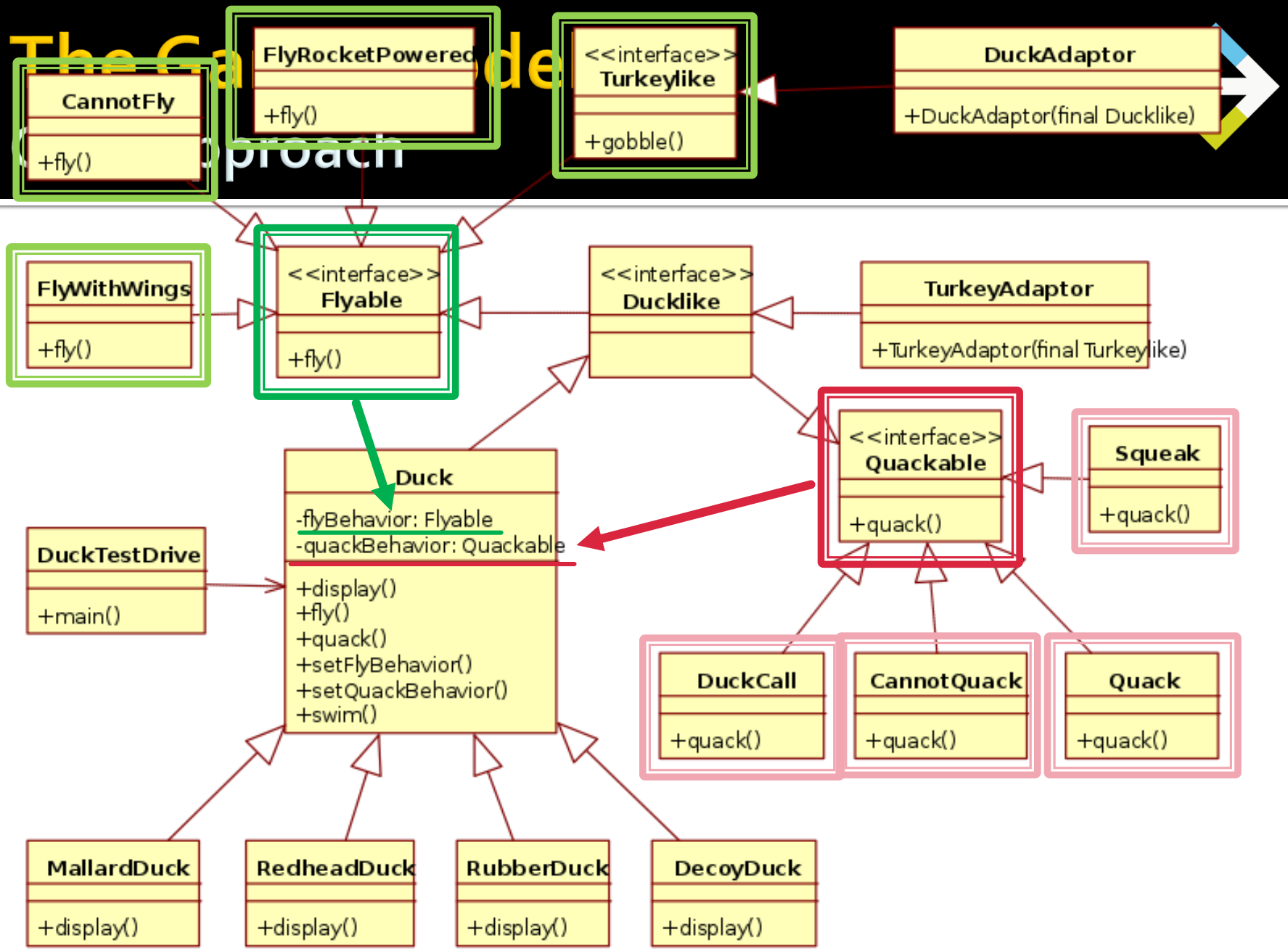
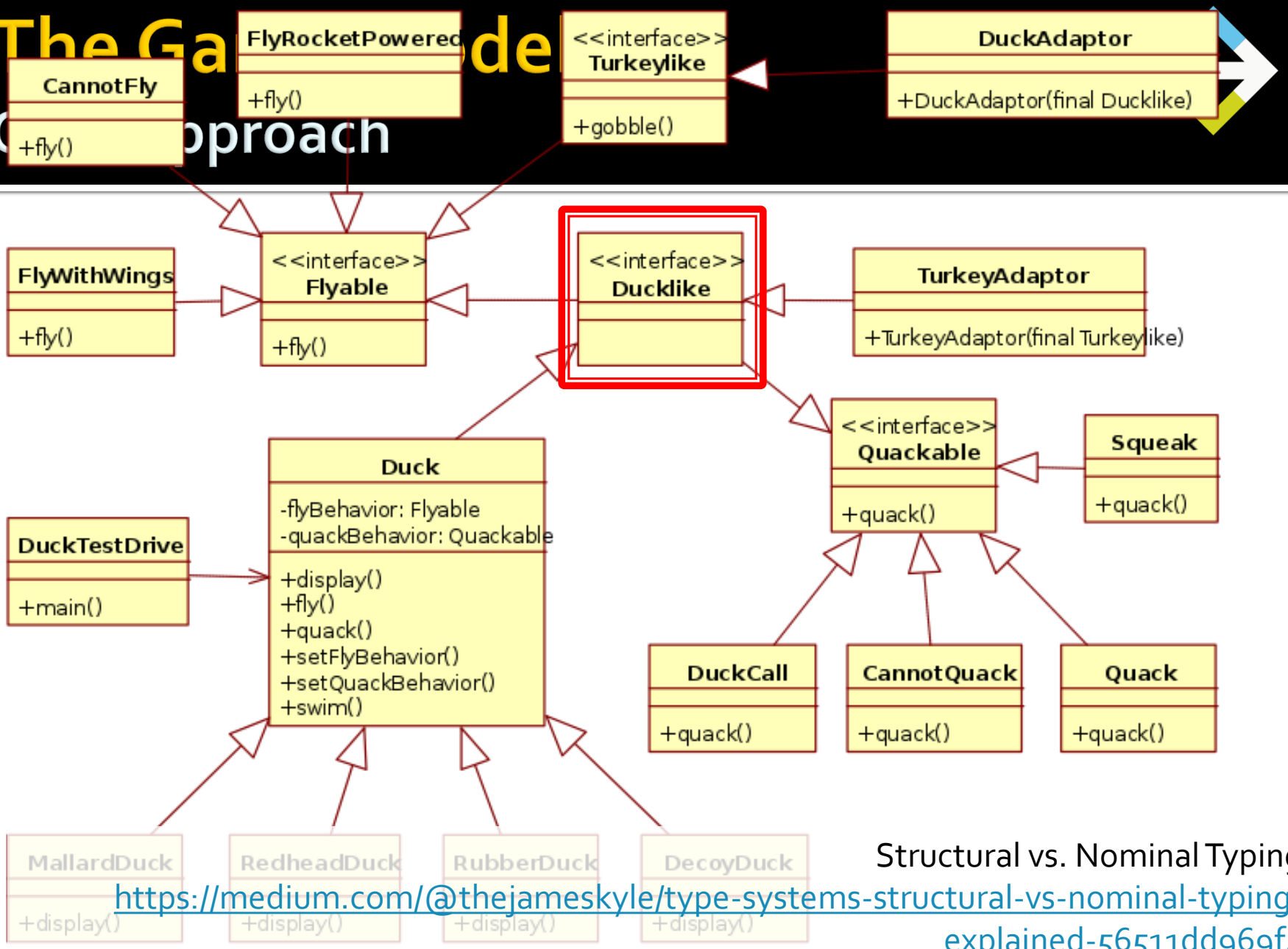


Image: https://en.wikipedia.org/wiki/File:UML_diagram_of_composition_over_inheritance.svg

The Ga de
C pproach



Structural vs. Nominal Typing
https://medium.com/@thejameskyle/type-systems-structural-vs-nominal-typing-explained-56511dd969f4

Image: https://en.wikipedia.org/wiki/File:UML_diagram_of_composition_over_inheritance.svg

The Ga…de
C…pproach

Next day, a game designer storms into your office screaming "ALL RUBBER DUCKS MUST GET ROCKET LAUNCHERS, and we need handicapped ducks that cannot swim."

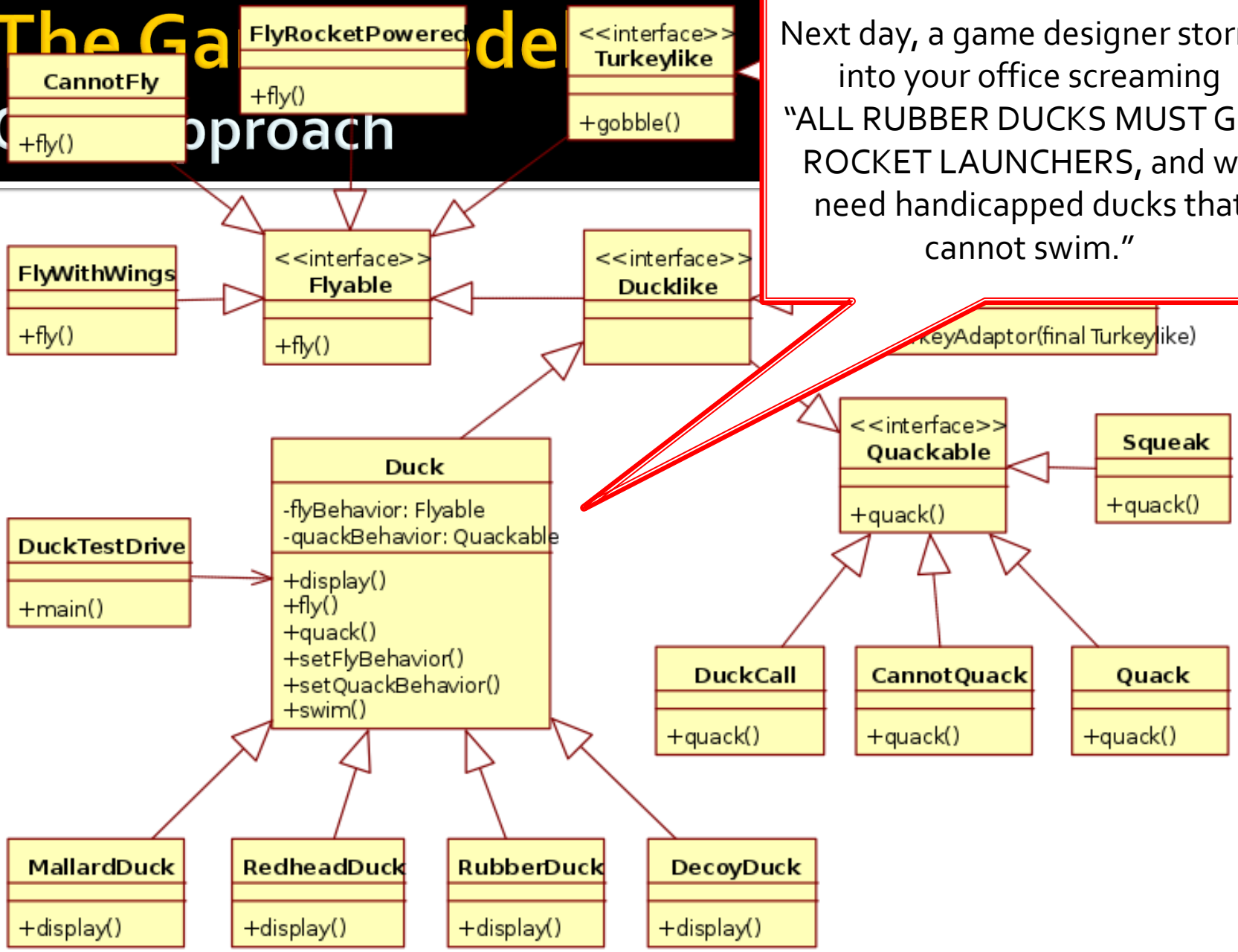


Image: https://en.wikipedia.org/wiki/File:UML_diagram_of_composition_over_inheritance.svg

The Game Object Model

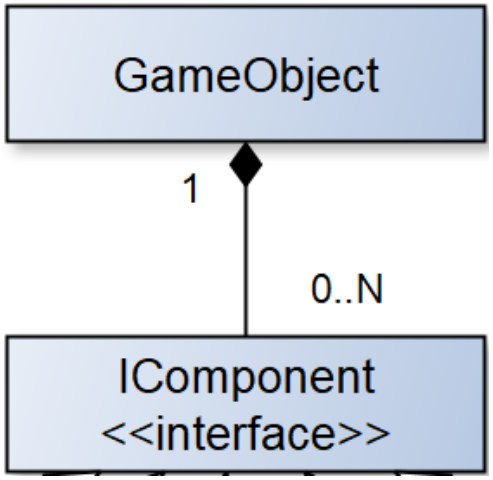
Component-based approach

ABSTRACTING GAME OBJECTS

… in a better way

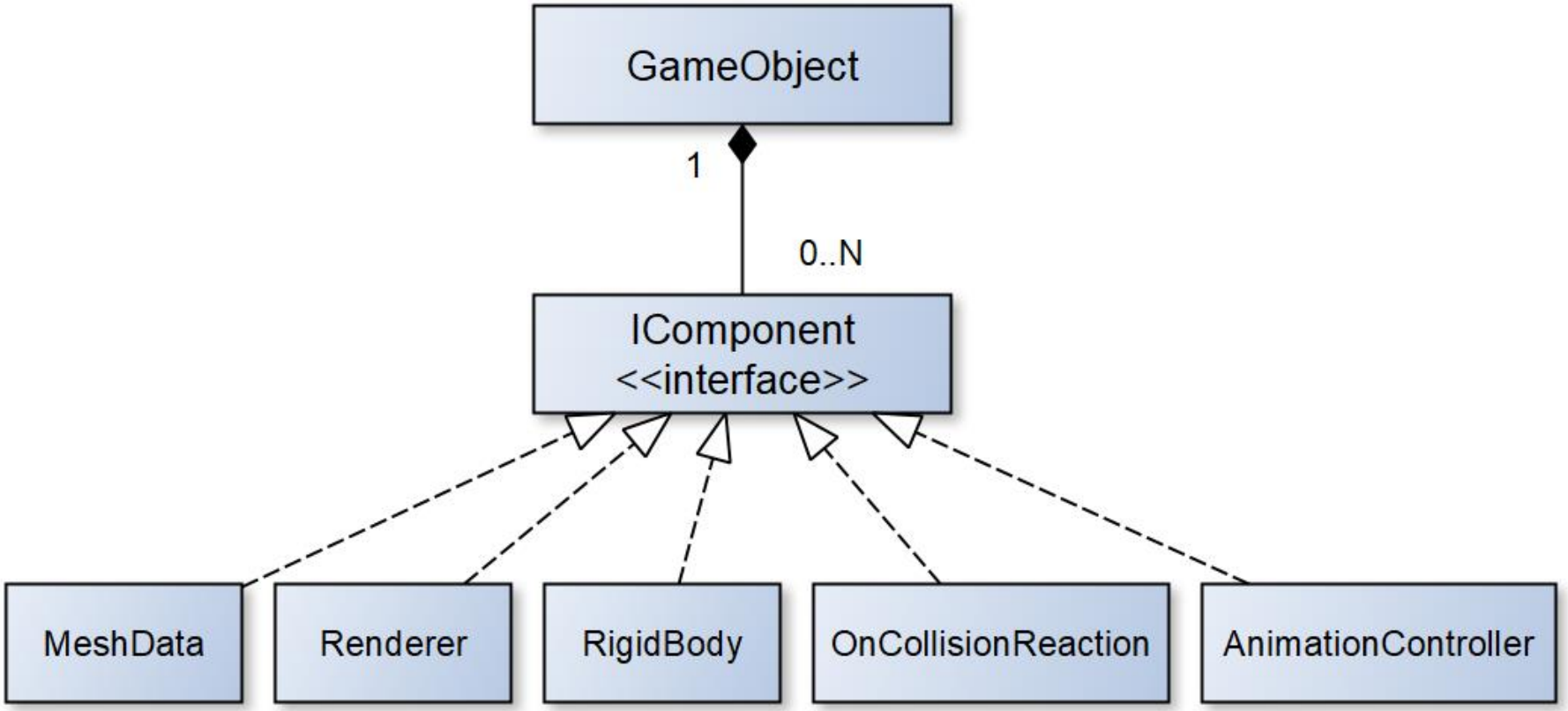
The Game Object Model

Component-based approach



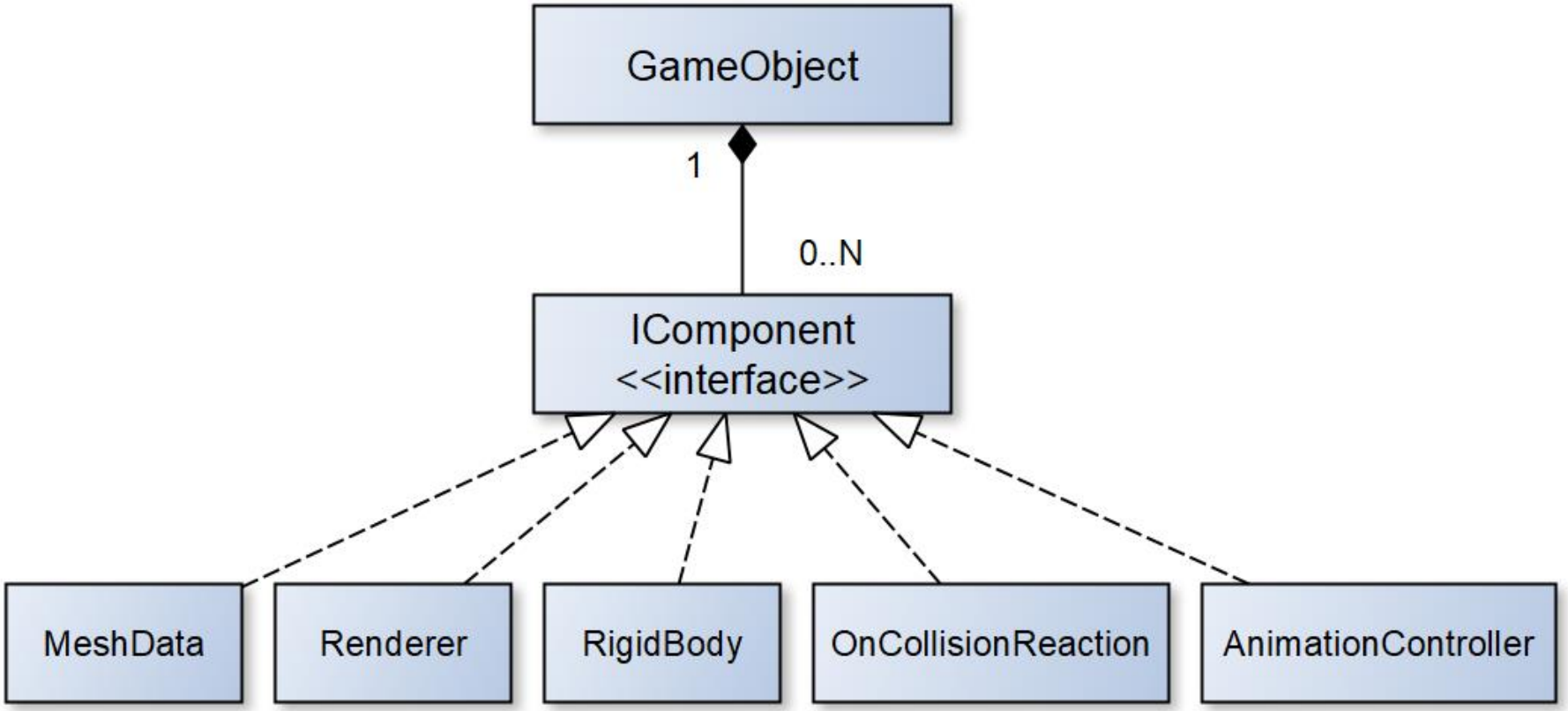
The Game Object Model

Component-based approach



The Game Object Model

Component-based approach



The Game Object Model

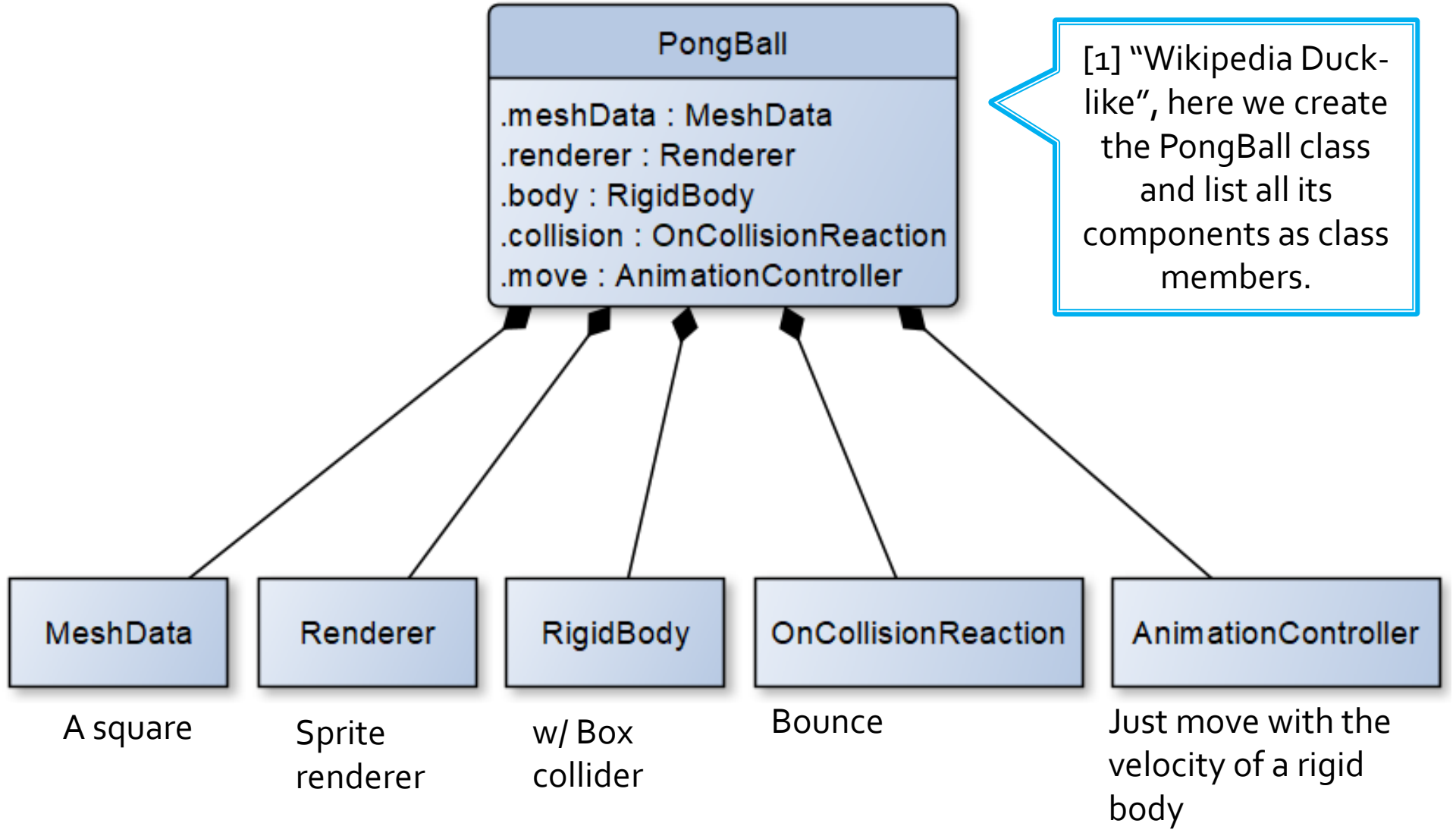
Component-based approach

HOW TO REPRESENT THE PONG BALL?

… there are three options

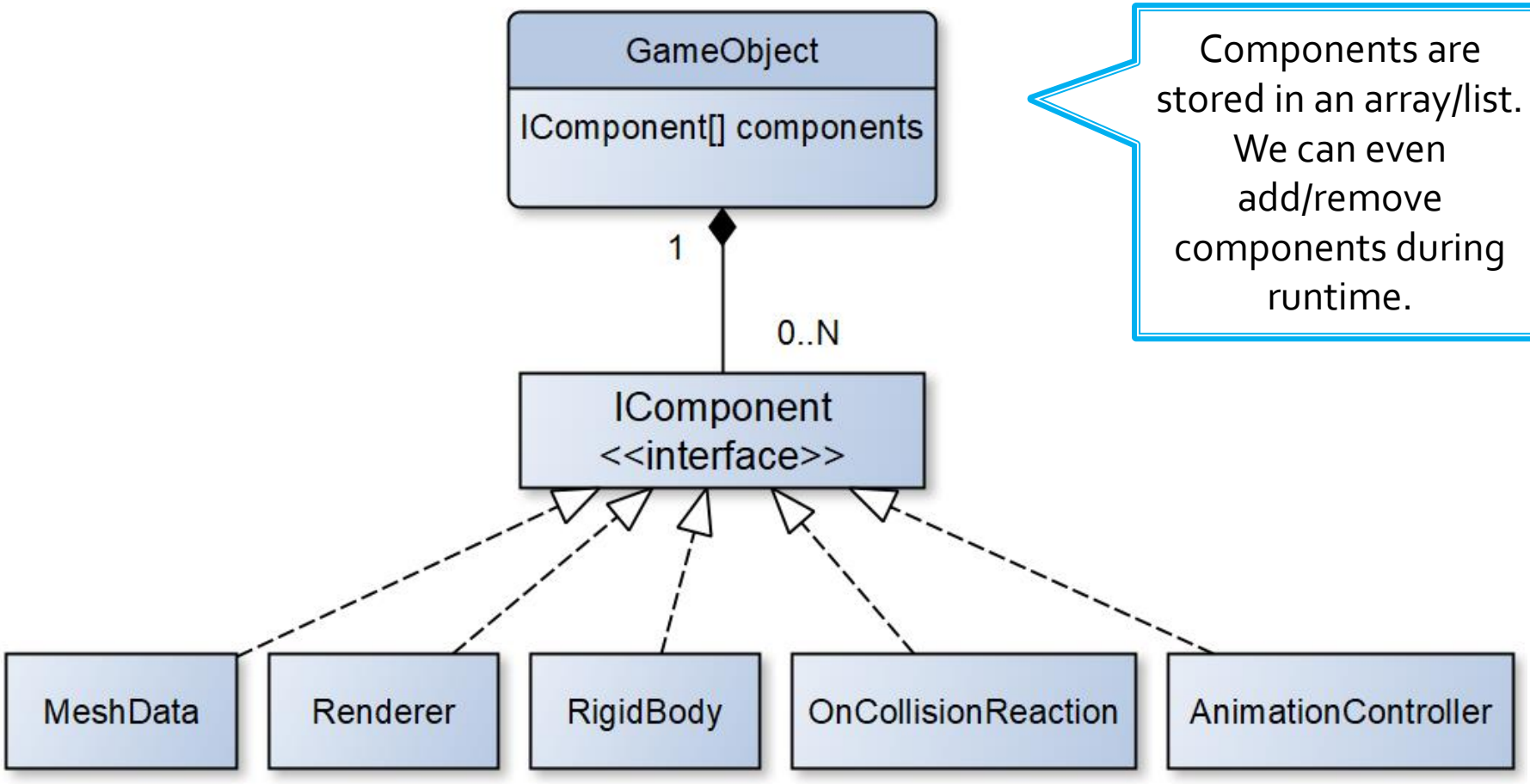
The Game Object Model

Component-based approach



The Game Object Model

Component-based approach

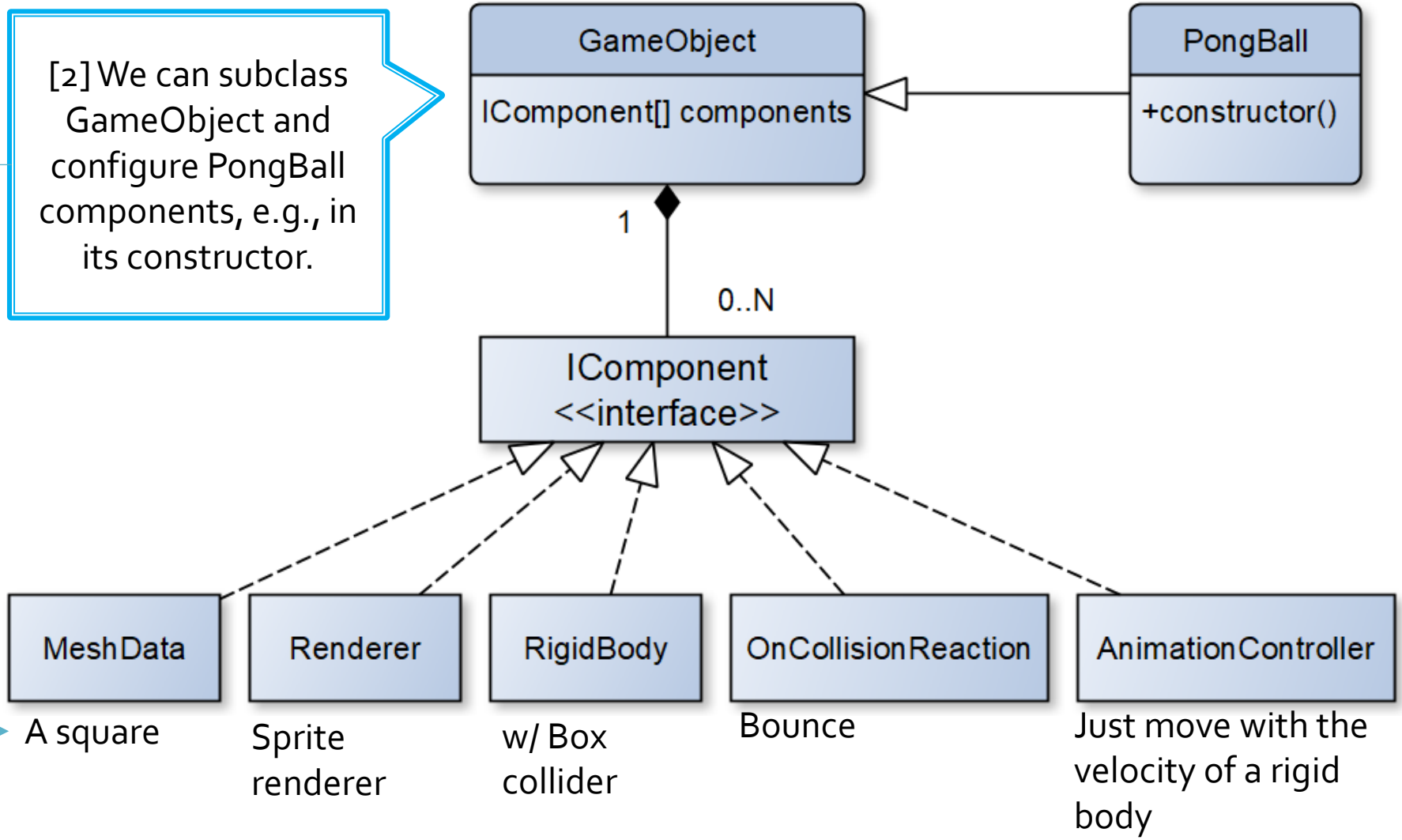


Components are stored in an array/list. We can even add/remove components during runtime.

The Game Object Model

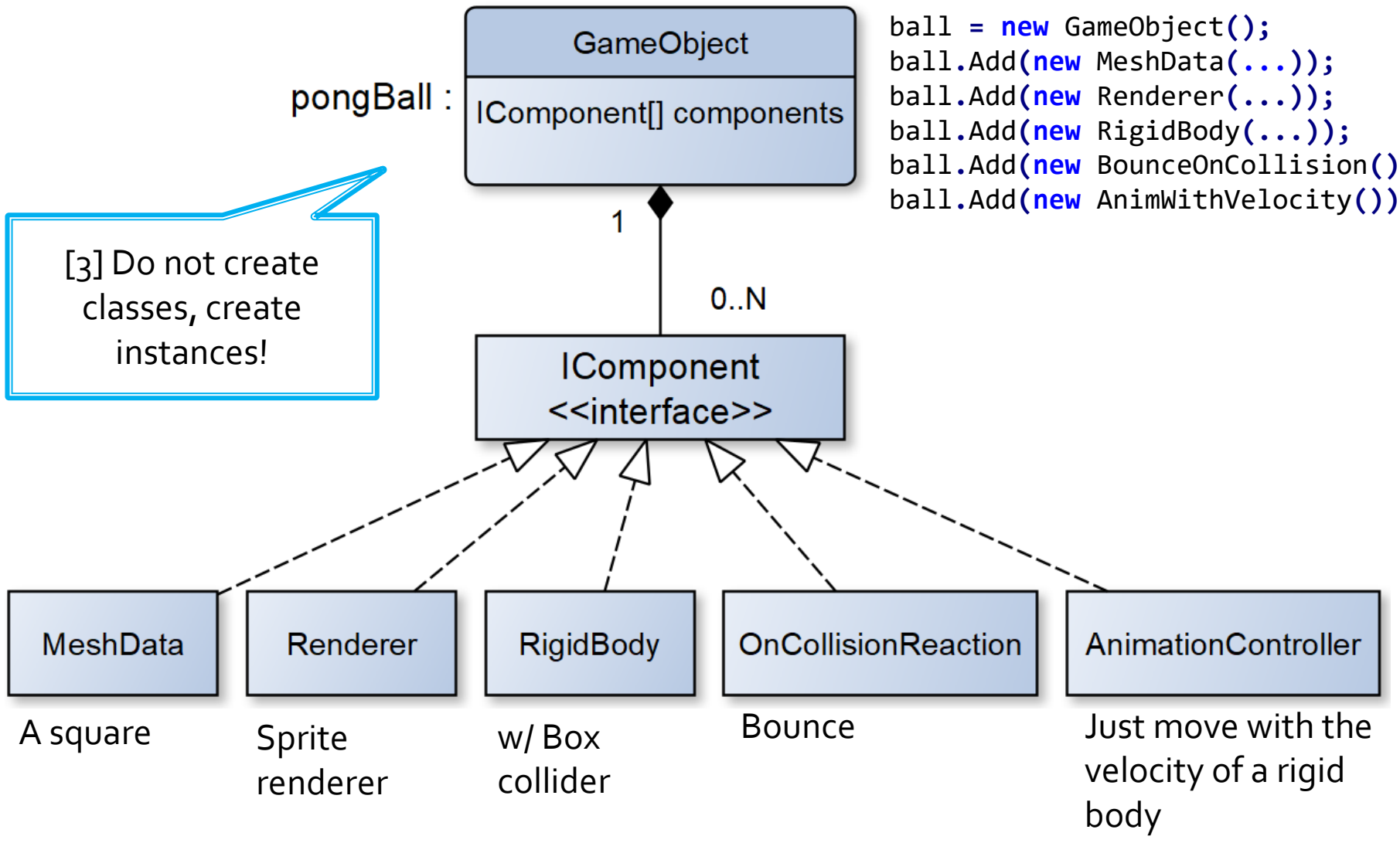
Component-based approach

[2] We can subclass GameObject and configure PongBall components, e.g., in its constructor.



The Game Object Model

Component-based approach

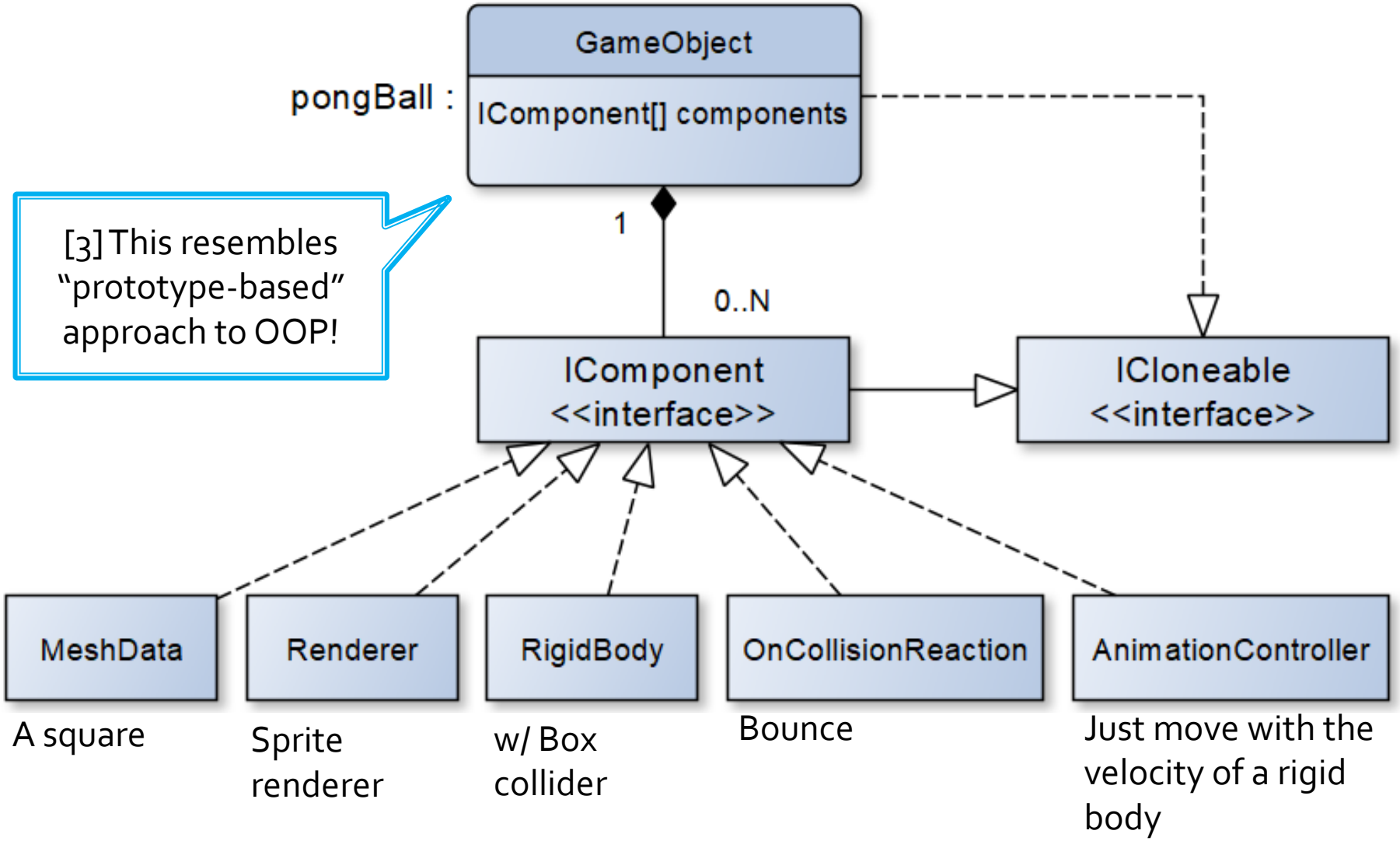



```
ball = new GameObject();
ball.Add(new MeshData(...));
ball.Add(new Renderer(...));
ball.Add(new RigidBody(...));
ball.Add(new BounceOnCollision());
ball.Add(new AnimWithVelocity());
```

[3] Do not create classes, create instances!

The Game Object Model

Component-based approach



The Scene Graph

Component-based approach

Describing the game world

The Scene Graph

onent-based approach

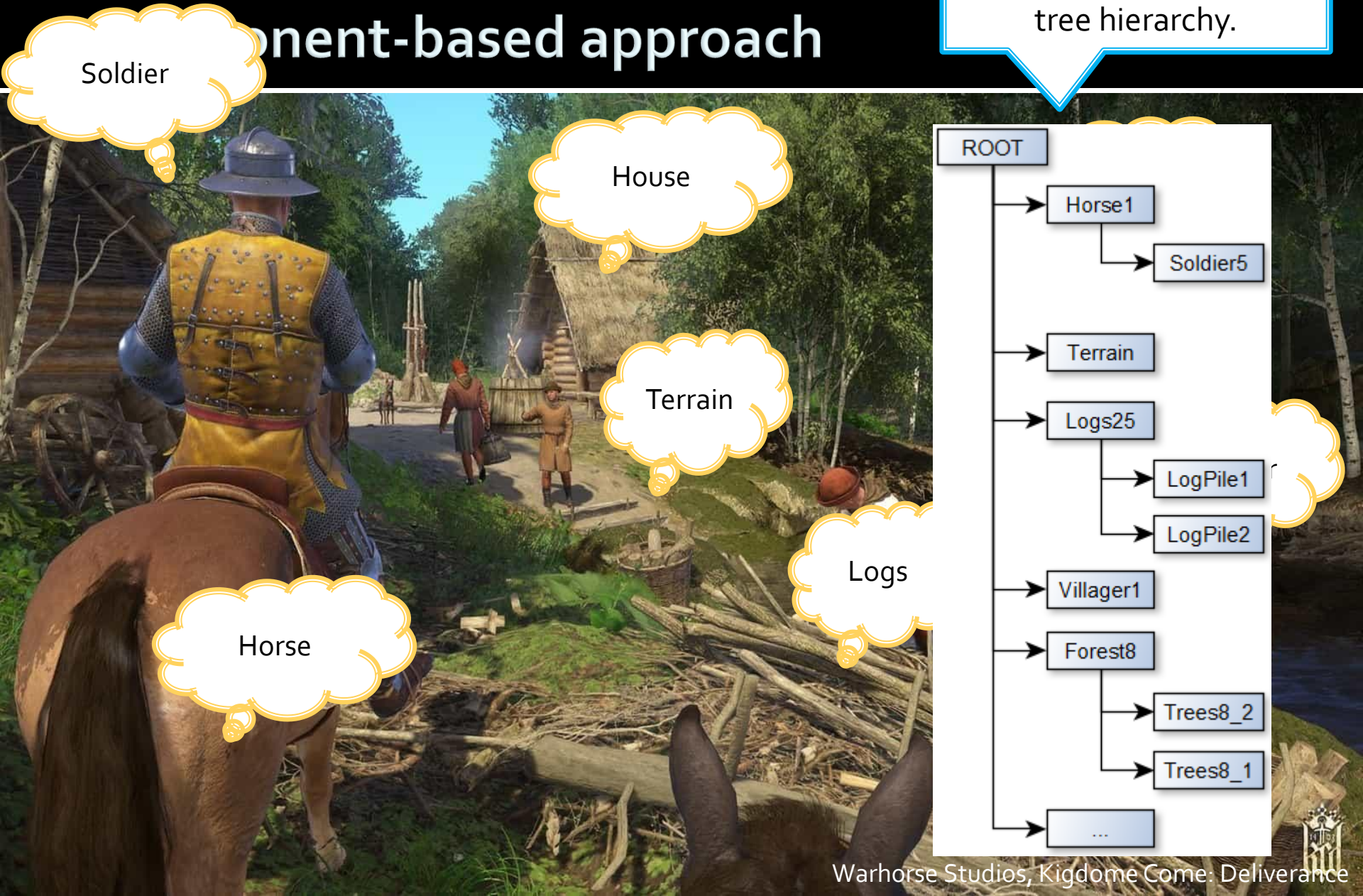
Objects making the scene are organized in a tree hierarchy.

Warhorse Studios, Kigdome Come: Deliverance

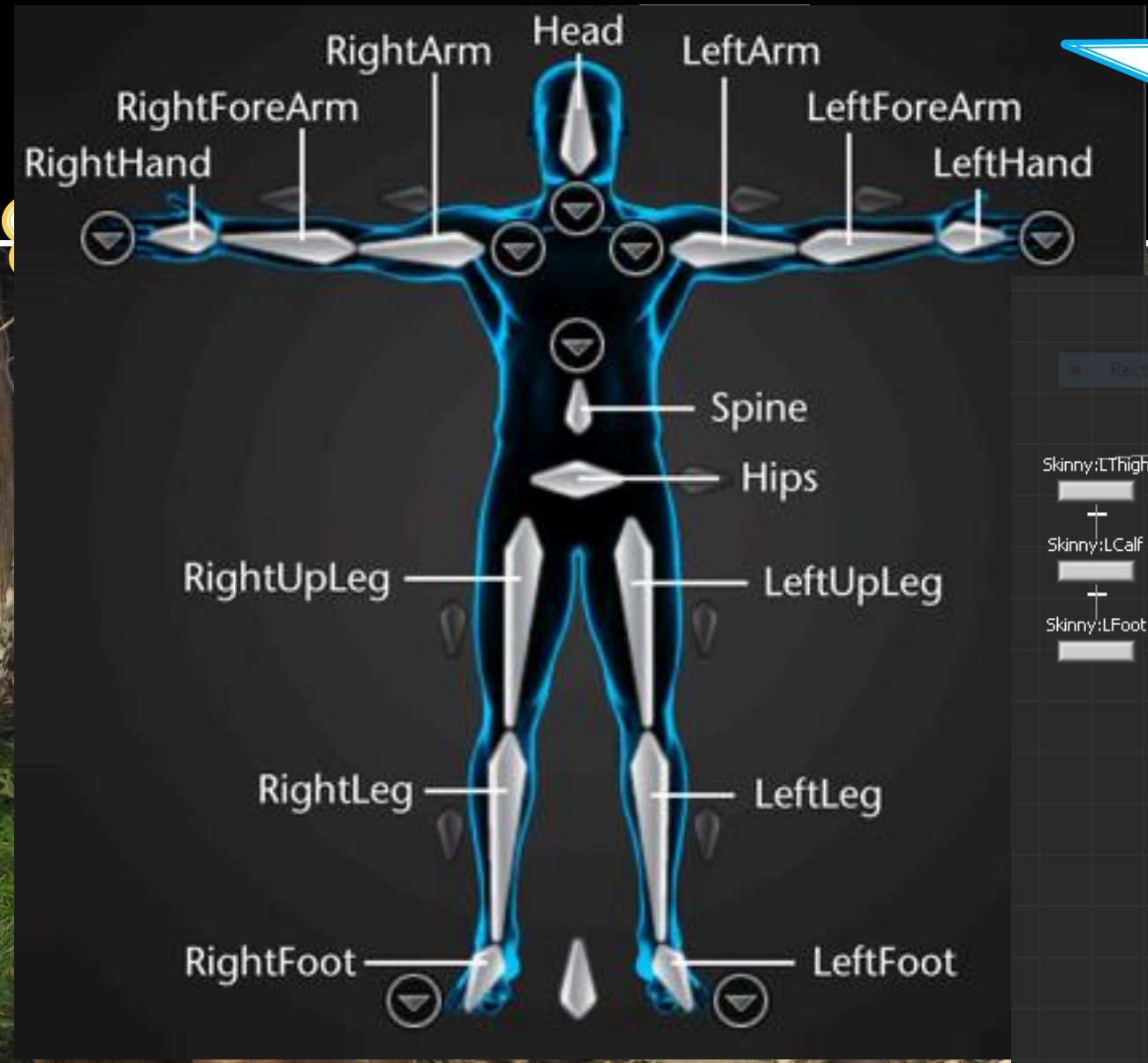
The Scene Graph

onent-based approach

Objects making the scene are organized in a tree hierarchy.



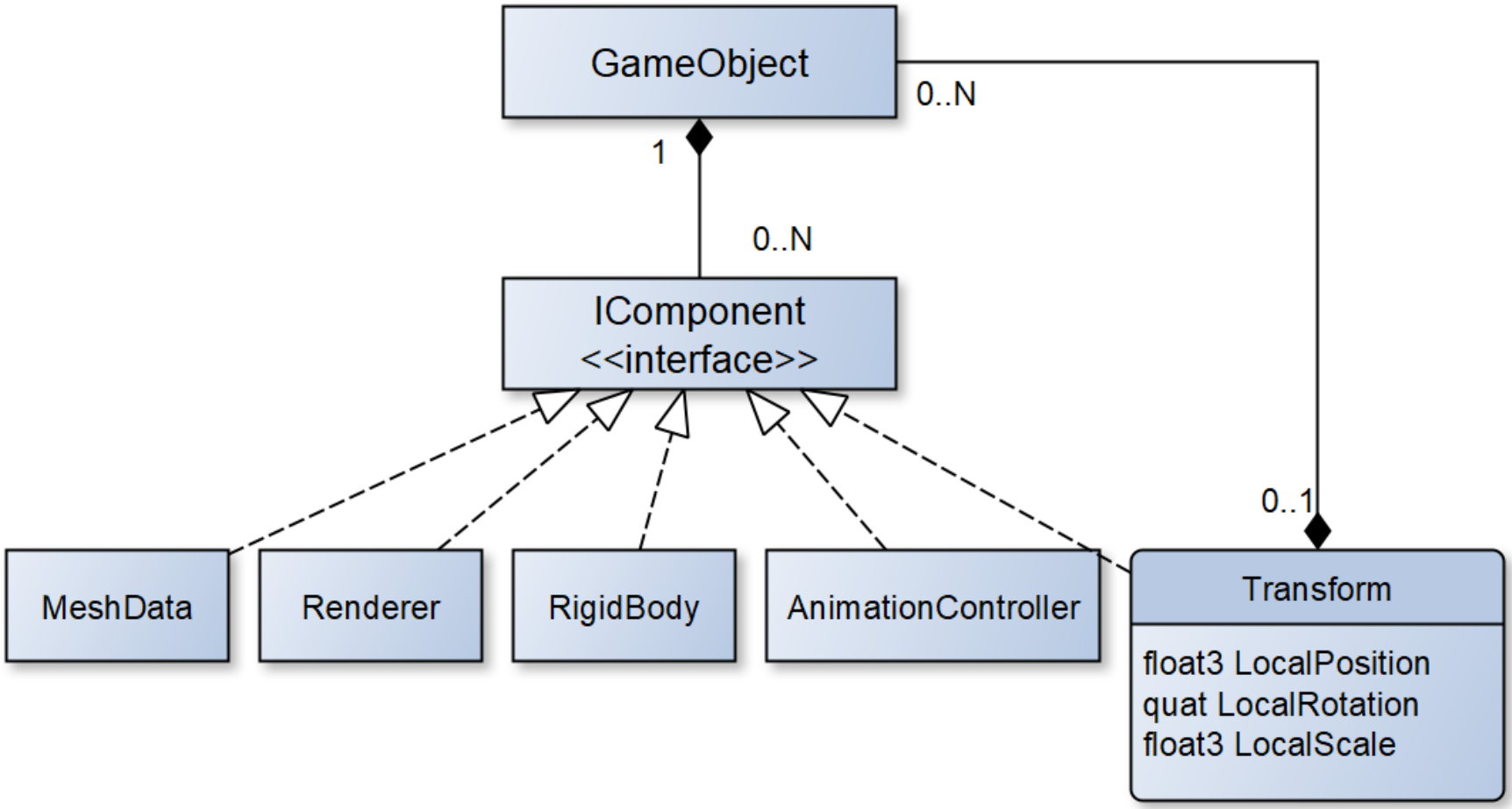
Warhorse Studios, Kigdome Come: Deliverance



Hierarchy can be deep, e.g., for the purpose of animations.

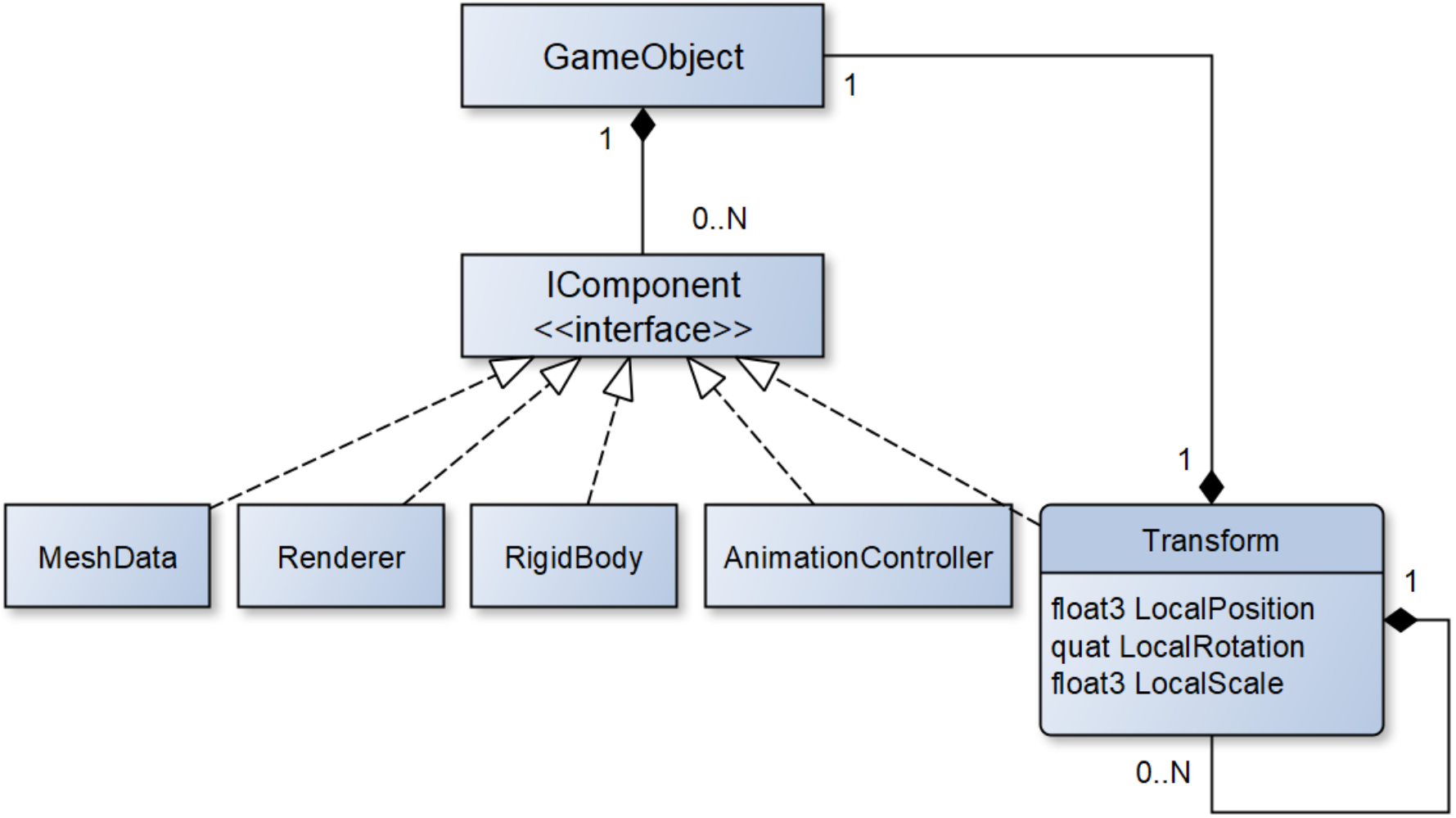
The Scene Graph

Component-based approach



The Scene Graph

Component-based approach



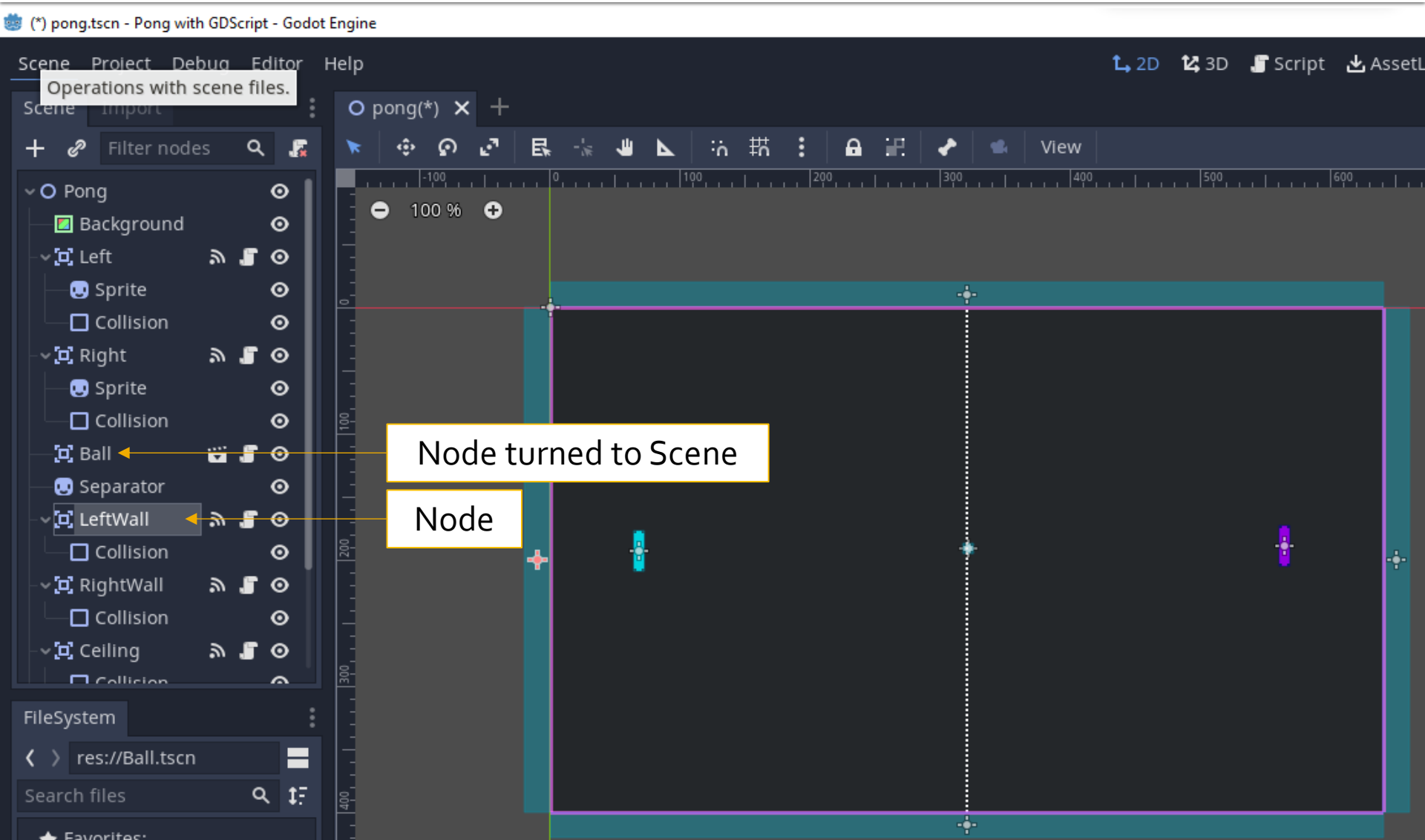
Game Object Model

The Godot Approach

GAME OBJECT == NODE or SCENE

Game Object Model

The Godot Approach



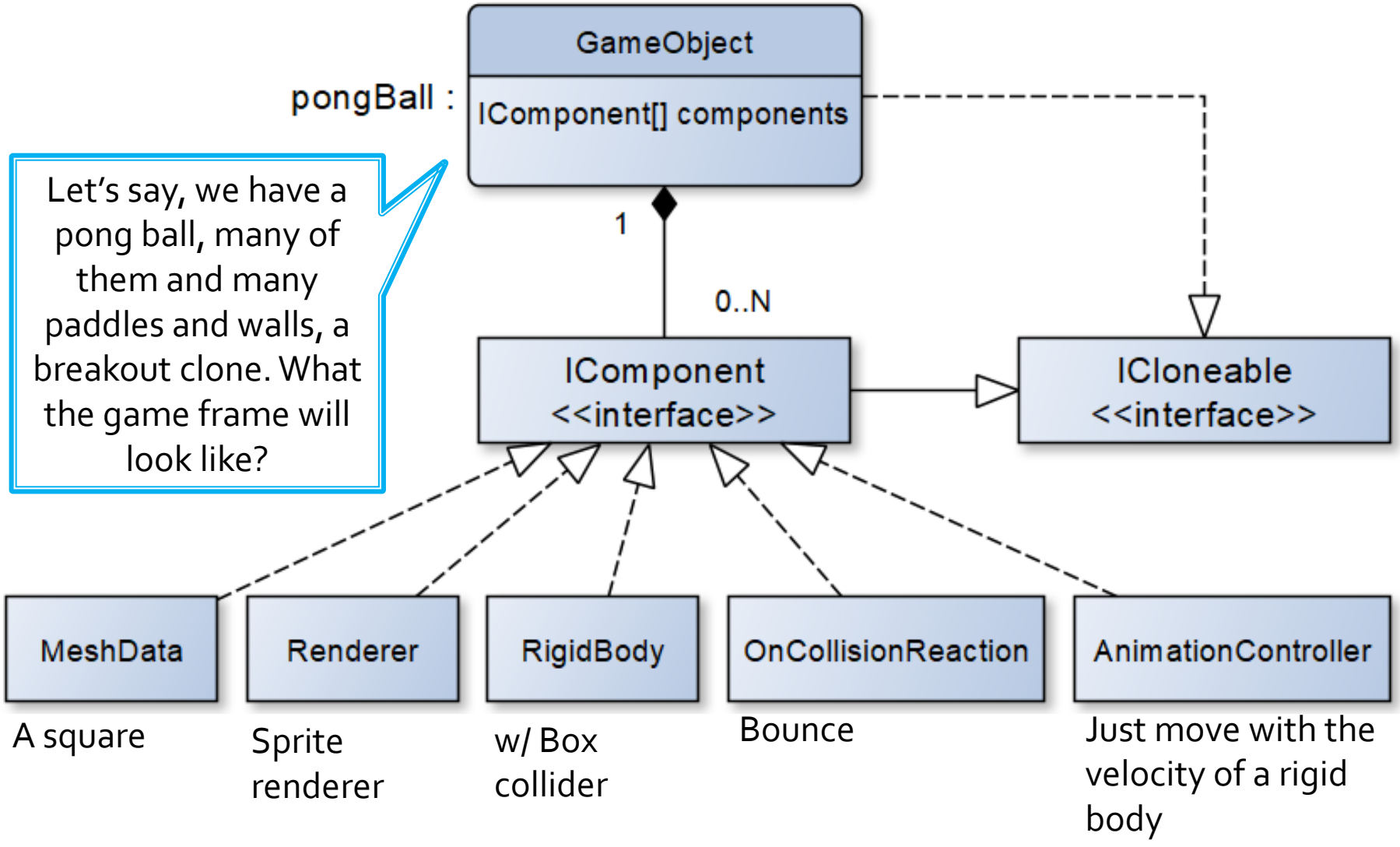
Game Object Model

Entity-Component-System approach

DO WE NEED THE GAME OBJECT CLASS?

Game Object Model

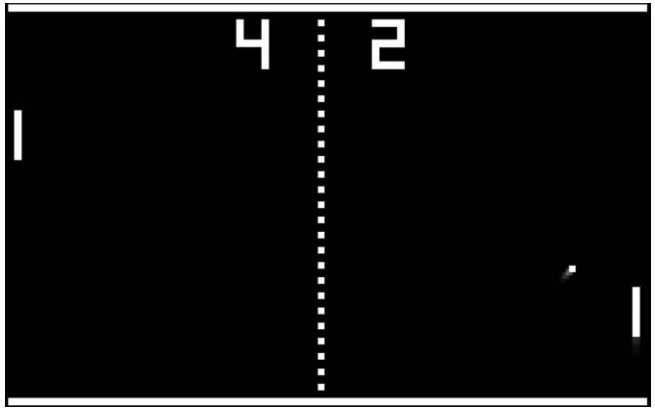
Entity-Component-System approach



Game Object Model

Entity-Component-System approach

```
void main() {
    initGame();
    while (true) {
        readPlayersInput();
        if (quitRequested()) break;
        movePaddles();
        moveBall();
        collideAndBounceBall();
        if (ballImpactedSide(LEFT_PLAYER)) {
            incremenentScore(RIGHT_PLAYER);
            resetBall();
        }
        else if (ballImpactedSide(RIGHT_PLAYER)) {
            incrementScore(LEFT_PLAYER);
            resetBall();
        }
        renderPlayfieldAndScores();
    }
}
```



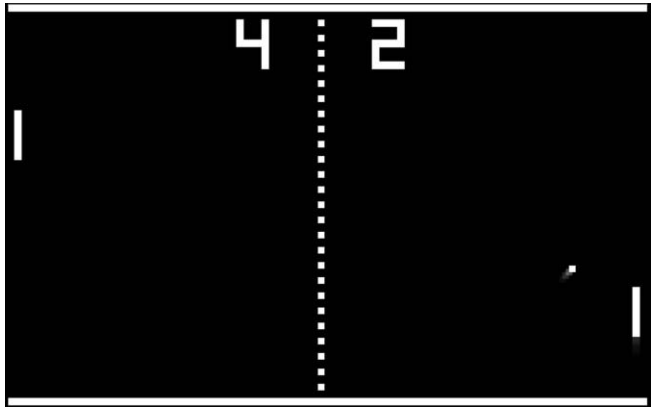
That's the game code for the Pong game we started with. What it will look like with dedicated subsystems?

Code adapted from: Gregory, J. (2018). Game engine architecture. CRC Press.

Game Object Model

Entity-Component-System approach

```
void main() {
    initGame();
    while (true) {
        inputSystem.Update();
        if (quitButtonPressed()) break;
        gameObjects.Update();
        physicsSystem.Update();
        animationSystem.Update();
        renderingSystem.Render();
    }
}
```



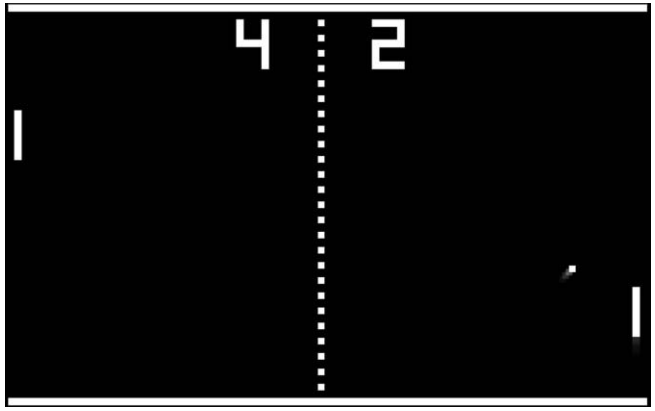
All the gameplay code is hidden in respective game object components, which are notified by respective systems if needed (e.g. collisions), or hidden in gameplay components that are being updated every frame (via `gameObjects.Update()`).

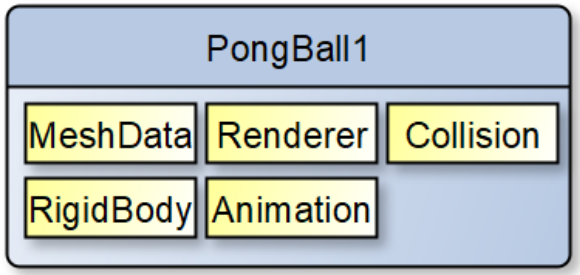
What implications an OOP component-based game object model have on the performance?

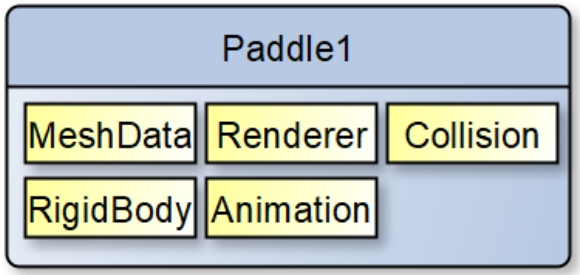
Game Object Model

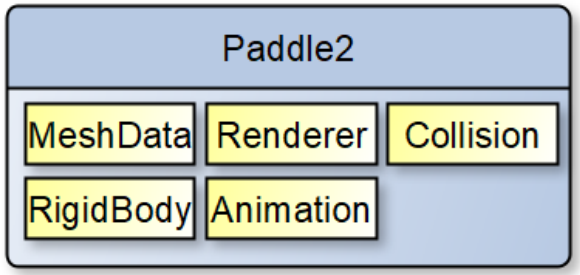
Entity-Component-System approach

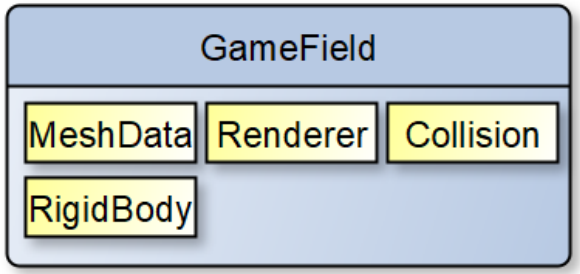
In OOP world, we tend to think about objects like this, nicely encapsulating its components.

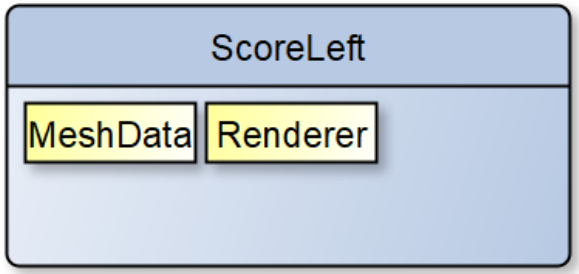


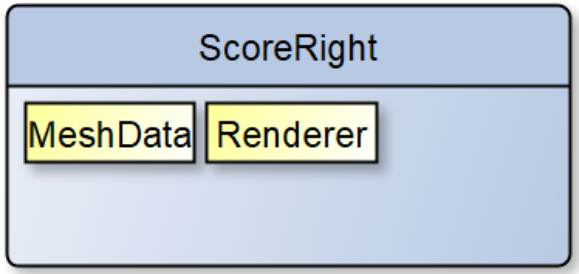






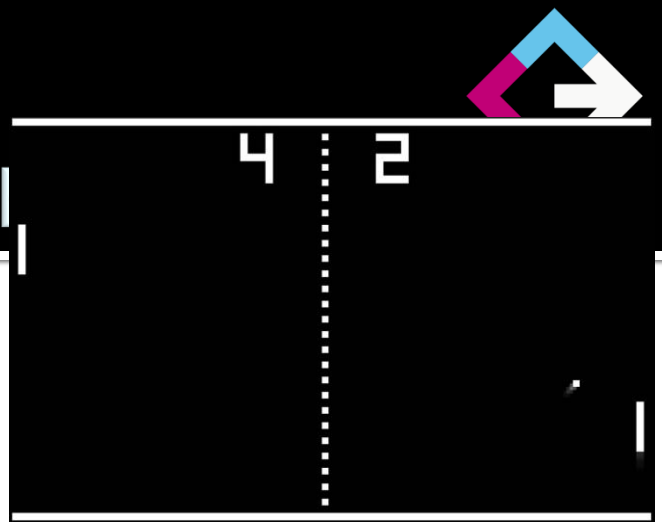




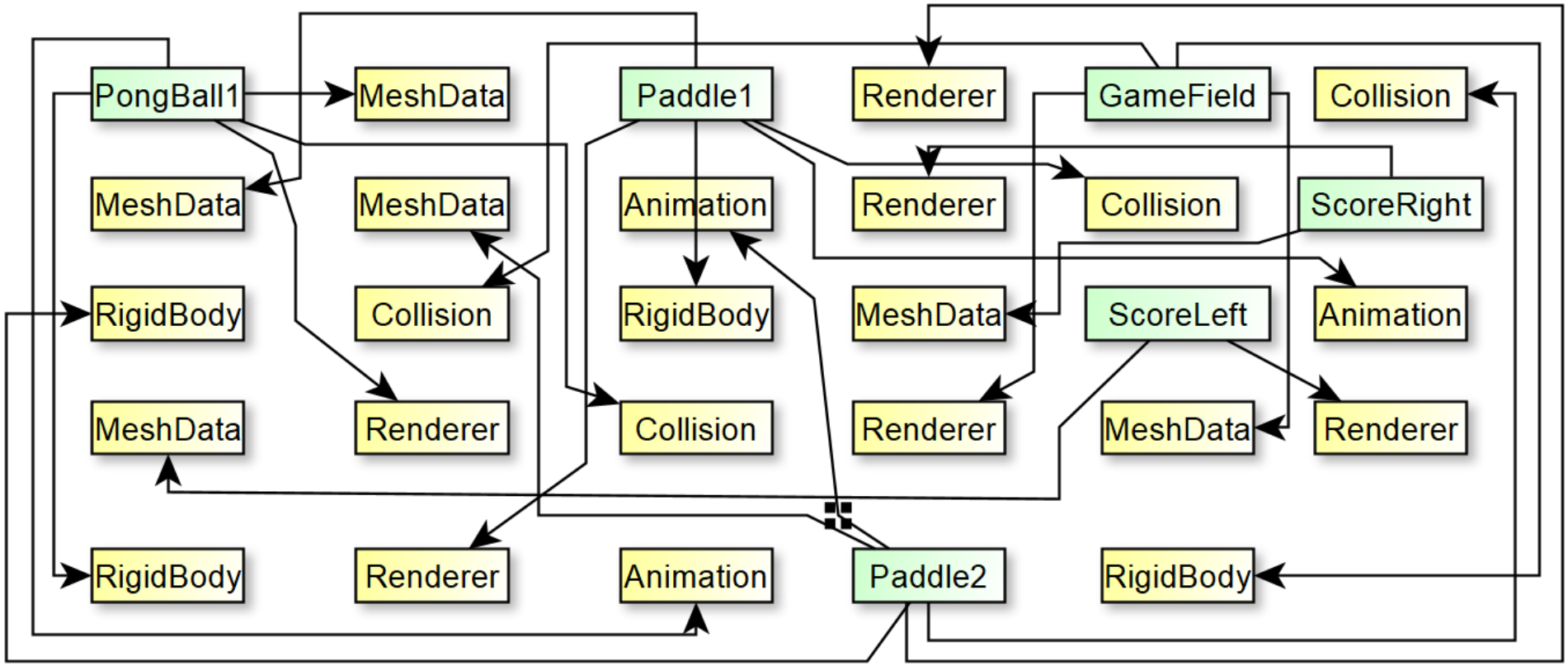


Game Object Model

Entity-Component-System ap

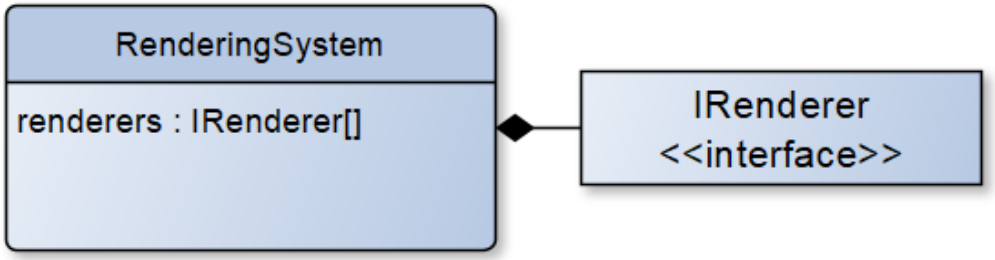


But in the **memory,** objects are of different sizes and their data will be spread throughout the memory.

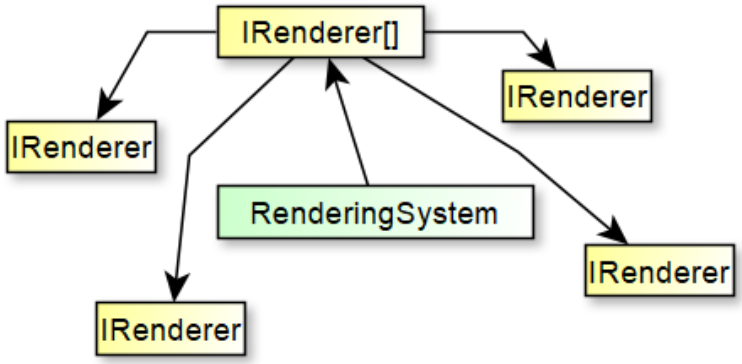


Game Object Model

Entity-Component-System approach



Memory




```
RenderingSystem.Render() {
    foreach (renderer in renderers) {
        renderer.Render()
    }
}
```

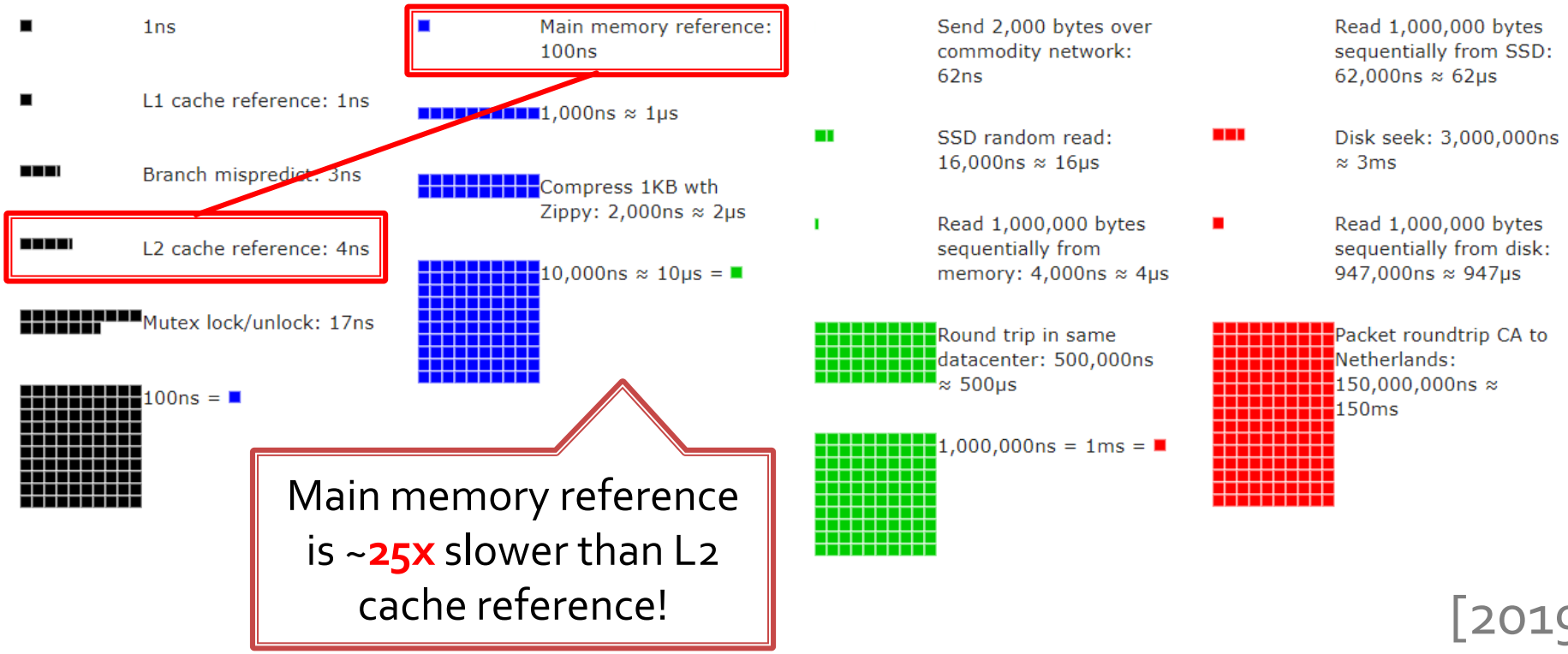
So when for instance rendering system will be doing its stuff with renders, it will work with data all over the memory. (Of course, reality is a bit different, but you should get the idea...)

This implies a lot of cache misses at the side of CPU!

Game Object Model
Entity-Component-System approach

There are some numbers associated you should have in mind!

1ns = 10^2x => 100 ns = 10^3x => 10µs = 10^6x => 1ms



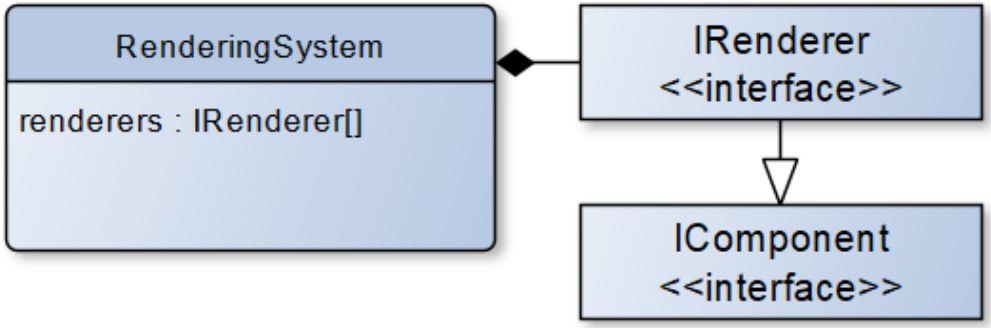
[2019]

Game Object Model

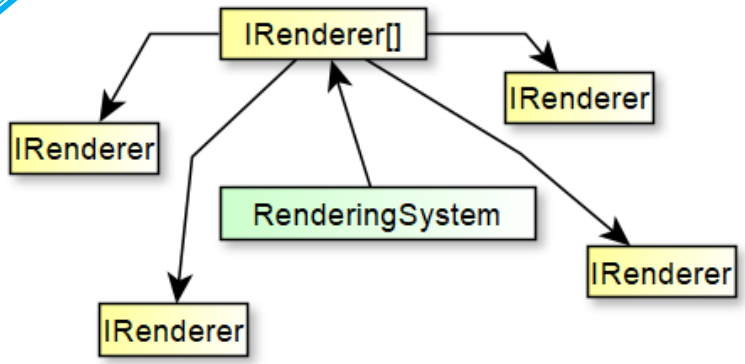
Entity-Component-System approach

1. We need to turn this…

Design



Memory

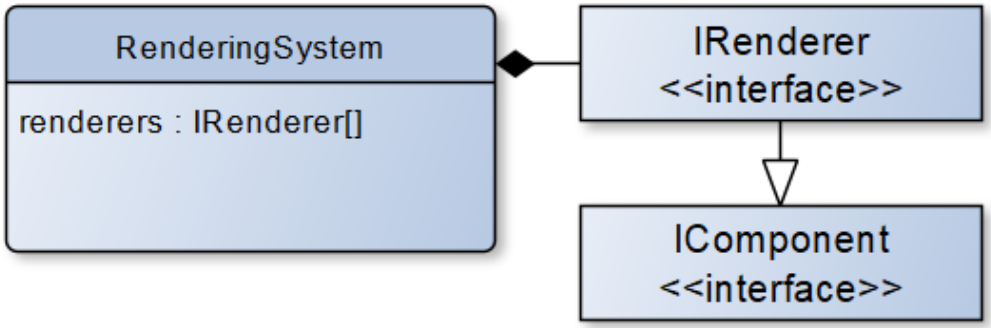


Game Object Model

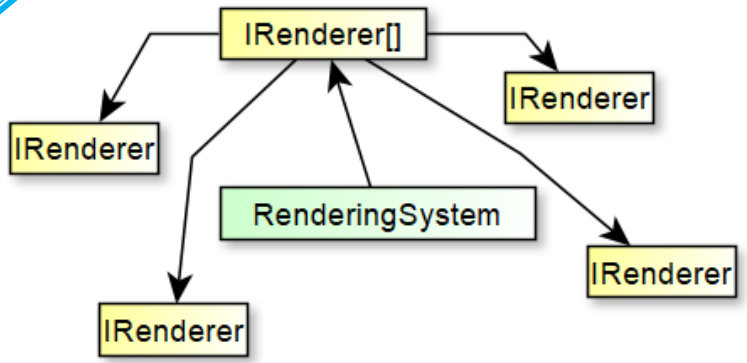
Entity-Component-System approach

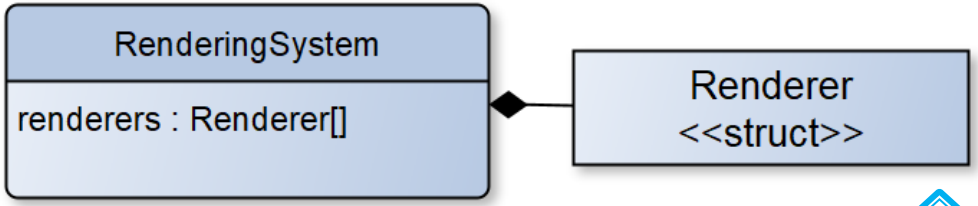
1. We need to turn this...

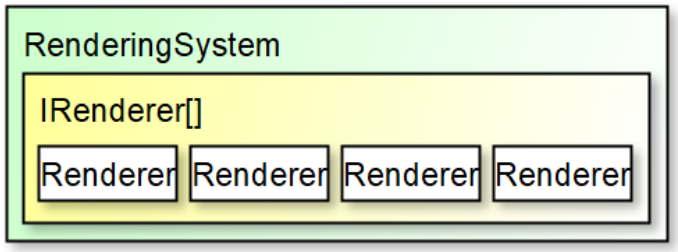
Design



Memory







2. E.g., into this

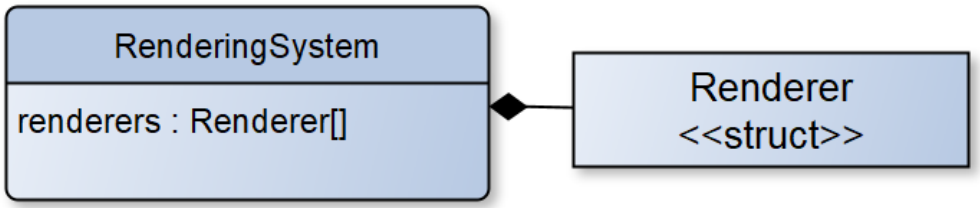
Game Object Model

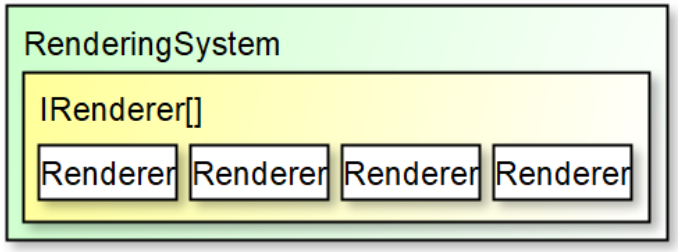
Entity-Component-System approach

The loop stays the same…

But this time, data are stored sequentially in the memory allowing for much faster execution of the for loop (reality is a bit more complicated again, but I hope you get the idea).

```
RenderingSystem.Render() {
    foreach (renderer in renderers) {
        renderer.Render()
    }
}
```

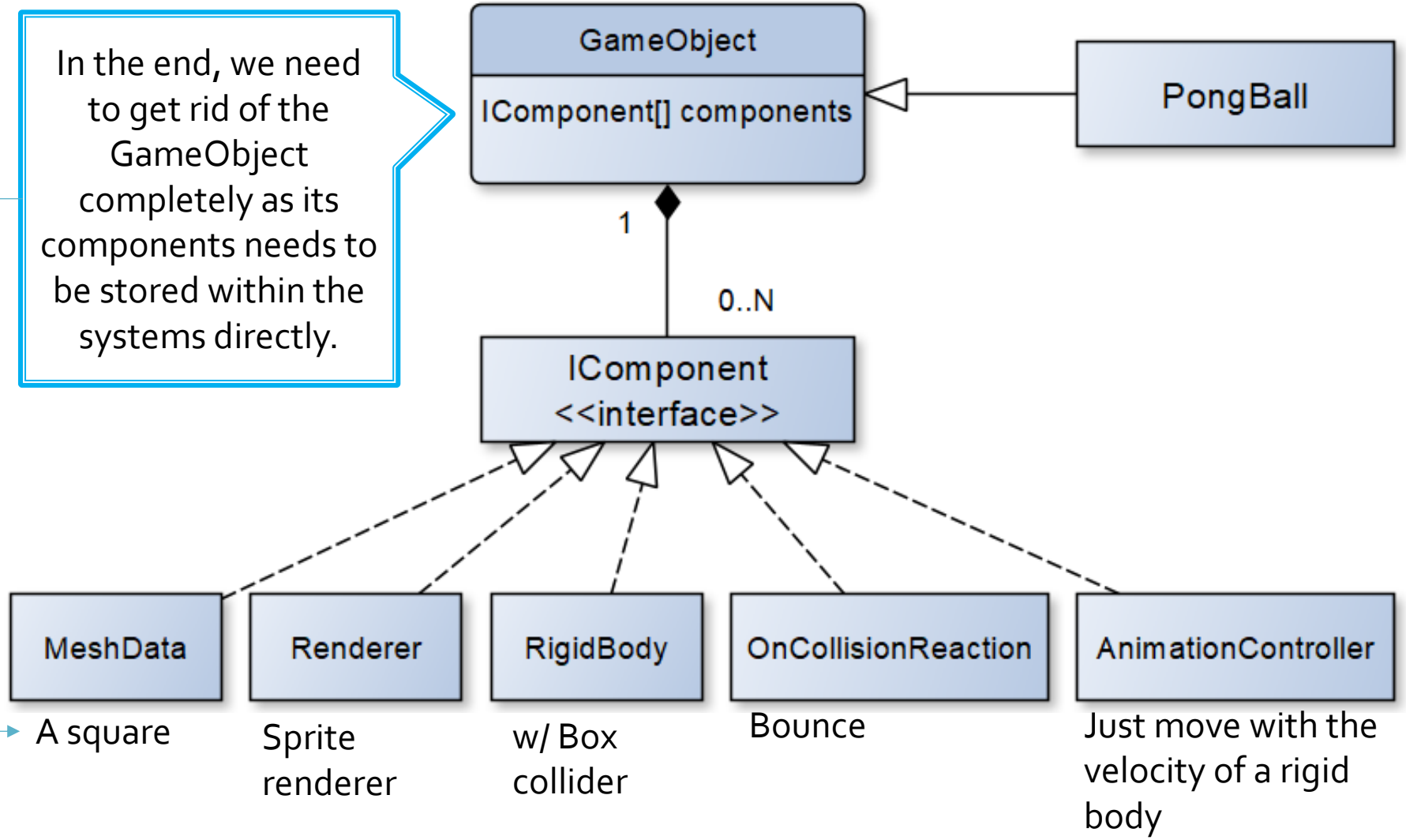




Game Object Model

Entity-Component-System approach

In the end, we need to get rid of the GameObject completely as its components needs to be stored within the systems directly.

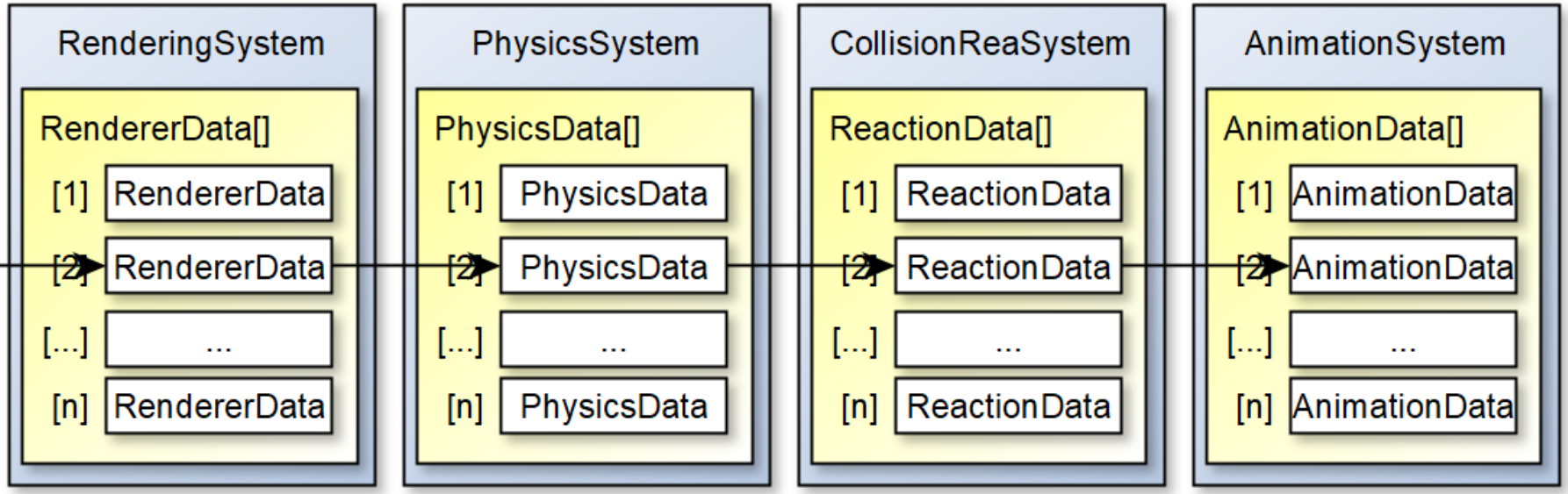


Game Object Model

Entity-Component-System approach

In ECS, a game object is identified by ID or set of IDs that (e.g.) matches their components in arrays of respective systems.

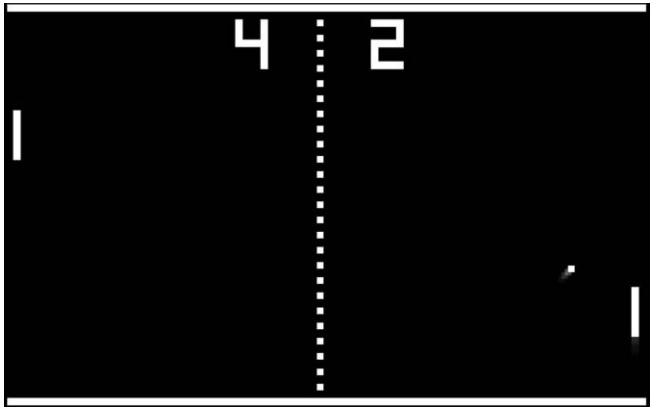
pongBall : int = 2



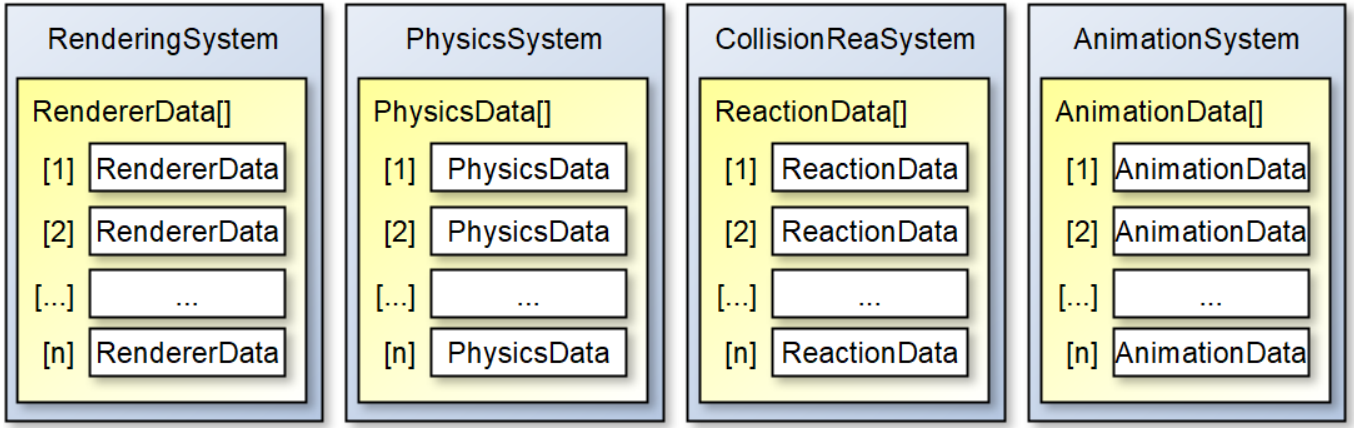
Game Object Model

Entity-Component-System approach

```
void main() {
    initGame();
    while (true) {
        inputSystem.Update();
        if (quitButtonPressed()) break;
        gameObjects.Update();
        physicsSystem.Update();
        animationSystem.Update();
        renderingSystem.Render();
    }
}
```



So when those systems updates, they can do it fast, it is easy to parallelize the computation and possibly to be run in parallel or jobified.



Game Object Model
Entity-Component-System approach

To recapitulate… in general, we split the world of game objects between:

1. Entities (each game object is tagged with Id)
2. Components (some Ids are associated with components, which consists of data only)
3. Systems (really the manipulator and drivers of the code)

"OOP is very good at implementing the part of any program that has lots of data … and … methods floating around which need to be executed only on a small, instanced, subset of the data in the program. OOP is very poor at implementing the "global" parts of a program, which have to operate "on everything", or have to be invoked "from everywhere". An ES solves this by explicitly dealing with all the global stuff using Systems, which are outside the realm of Entity/Component."

– *Adam Martin*, Entity Systems are the future of MMOG development – Part 2

The Game Model

Let's reminiscent…

The Game Model

Extra stuff

Game Object foundation, GDC 2002 talk by Scott Bilas
Link

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

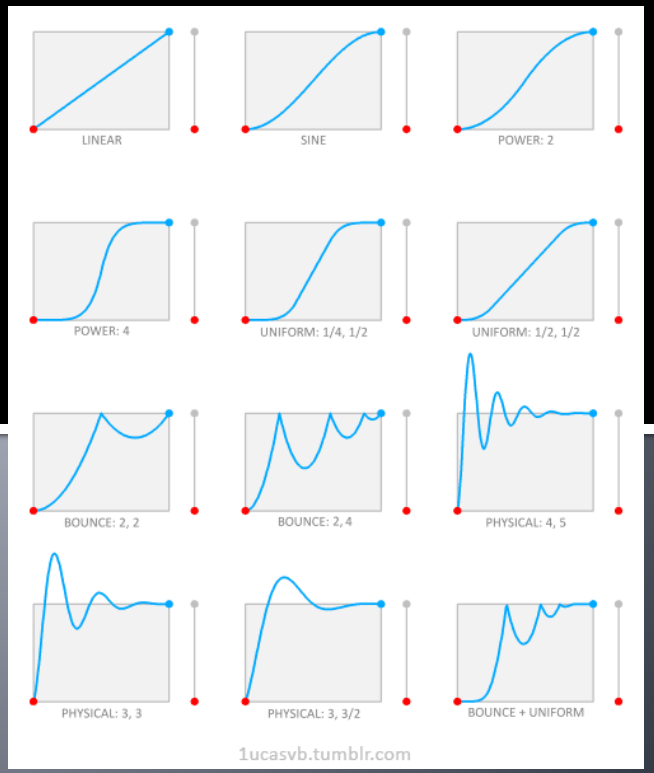
Material has been produced within and supported by the project „Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“ kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.

Faculty of Mathematics and Physics
Charles University, Prague
20th October 2023

Gameplay Programming Level 04

(In-Be)Tween(ing)
and Juiciness

Curvy (s)lin(m)es everywhere



Tweening!

There is so much stuff over the internet!

So we will not be reinventing
the wheel here with slides...

Tweening

Go and watch…

Secrets of Game Feel and Juice (len. 5:18)
https://www.youtube.com/watch?v=216_5nu4aVQ

Secrets of Game Feel and Juice

Wrap up

… an invisible art … something players will immediately detect …
… the juice is about doubling down what your game is about …

Playability

Emotion
Satisfaction
Learning
Efficiency
Immersion
Motivation
Socialization

Secrets of Game Feel and Juice

Wrap up

Super Mario GIF
on the next slide

Playability

Emotion
Satisfaction
Learning
Efficiency
Immersion
Motivation
Socialization

Secrets of Game Feel and Juice

Wrap up

Playability

Emotion
Satisfaction
Learning
Efficiency
Immersion
Motivation
Socialization

Secrets of Game Feel and Juice

Wrap up

Juice

== ? ==

Playability

Emotion
Satisfaction
Learning
Efficiency
Immersion
Motivation
Socialization

Secrets of Game Feel and Juice

Wrap up

… an invisible art … something players will immediately detect …
… the juice is about doubling down what your game is about …

Secrets of Game Feel and Juice

Wrap up

… an invisible art … something players will immediately detect …
… the juice is about doubling down what your game is about …

Secrets of Game Feel and Juice

Wrap up

… an invisible art … something players will immediately detect …
… the juice is about doubling down what your game is about …

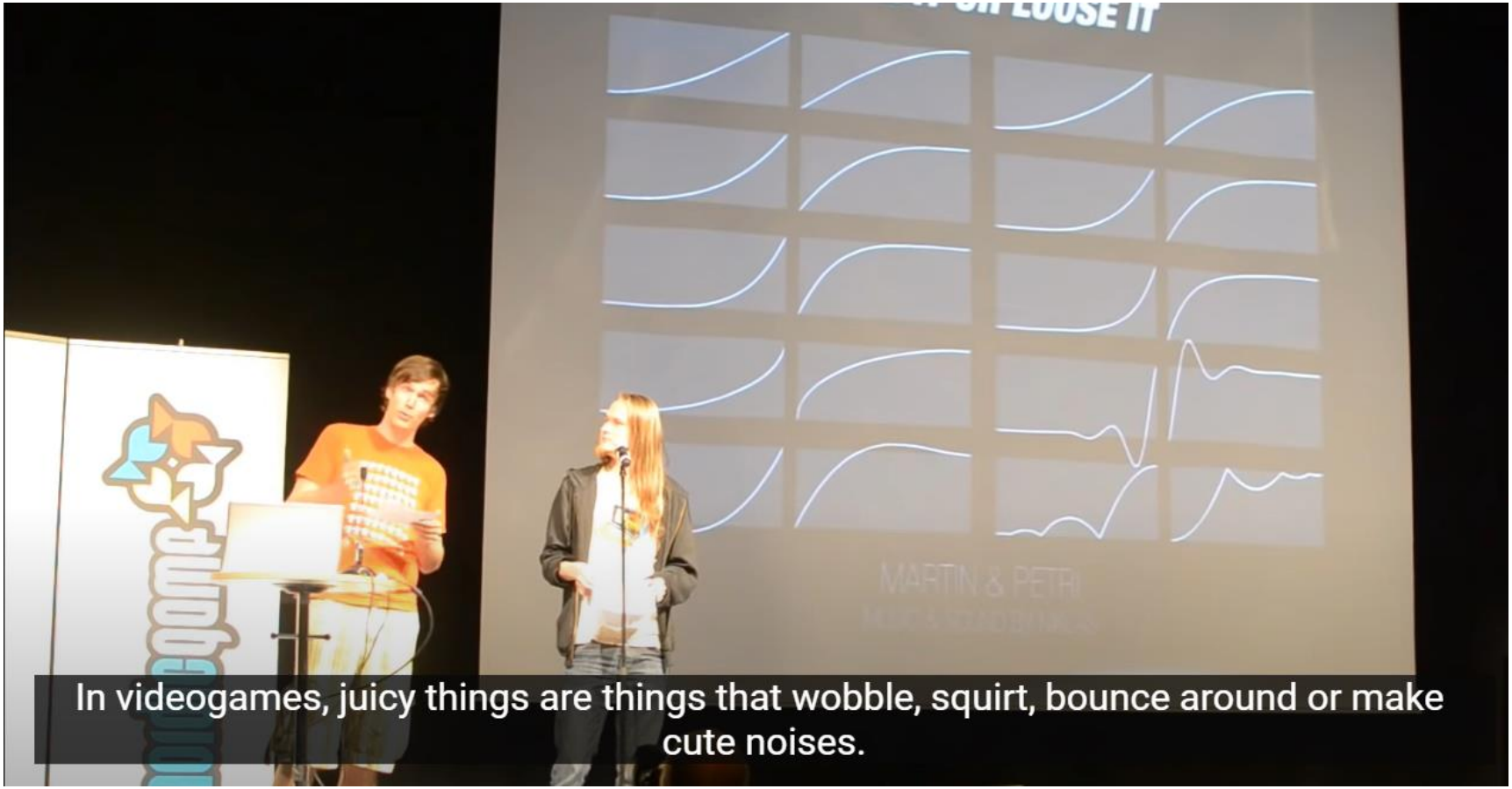
Secrets of Game Feel and Juice

Wrap up

… an invisible art … something players will immediately detect …
… the juice is about doubling down what your game is about …

Tweening

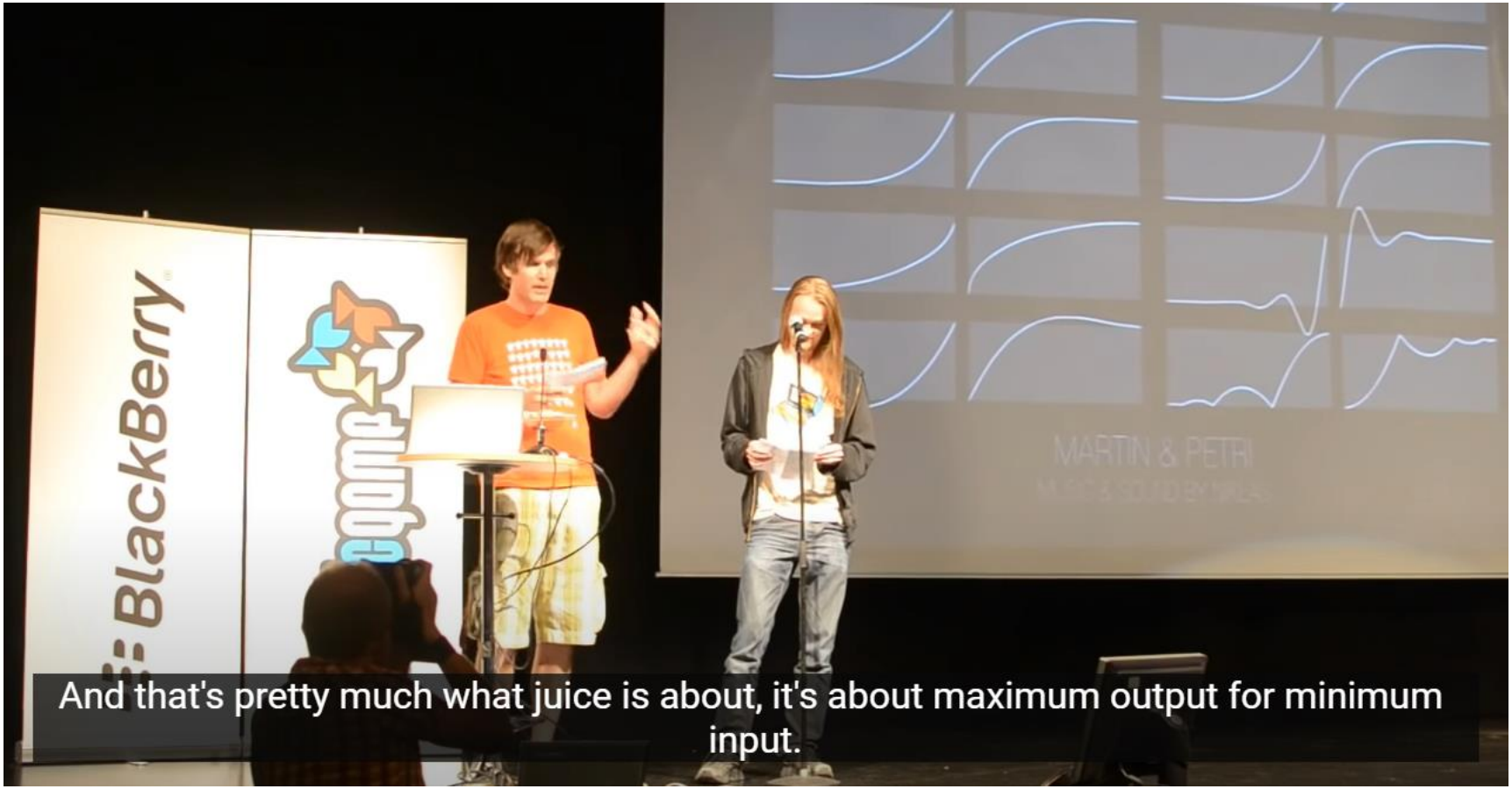
Go and watch…



Juice it or Lose it! [length 15:37]
https://www.youtube.com/watch?v=FyoaCDmgnxg&t=2s

Tweening

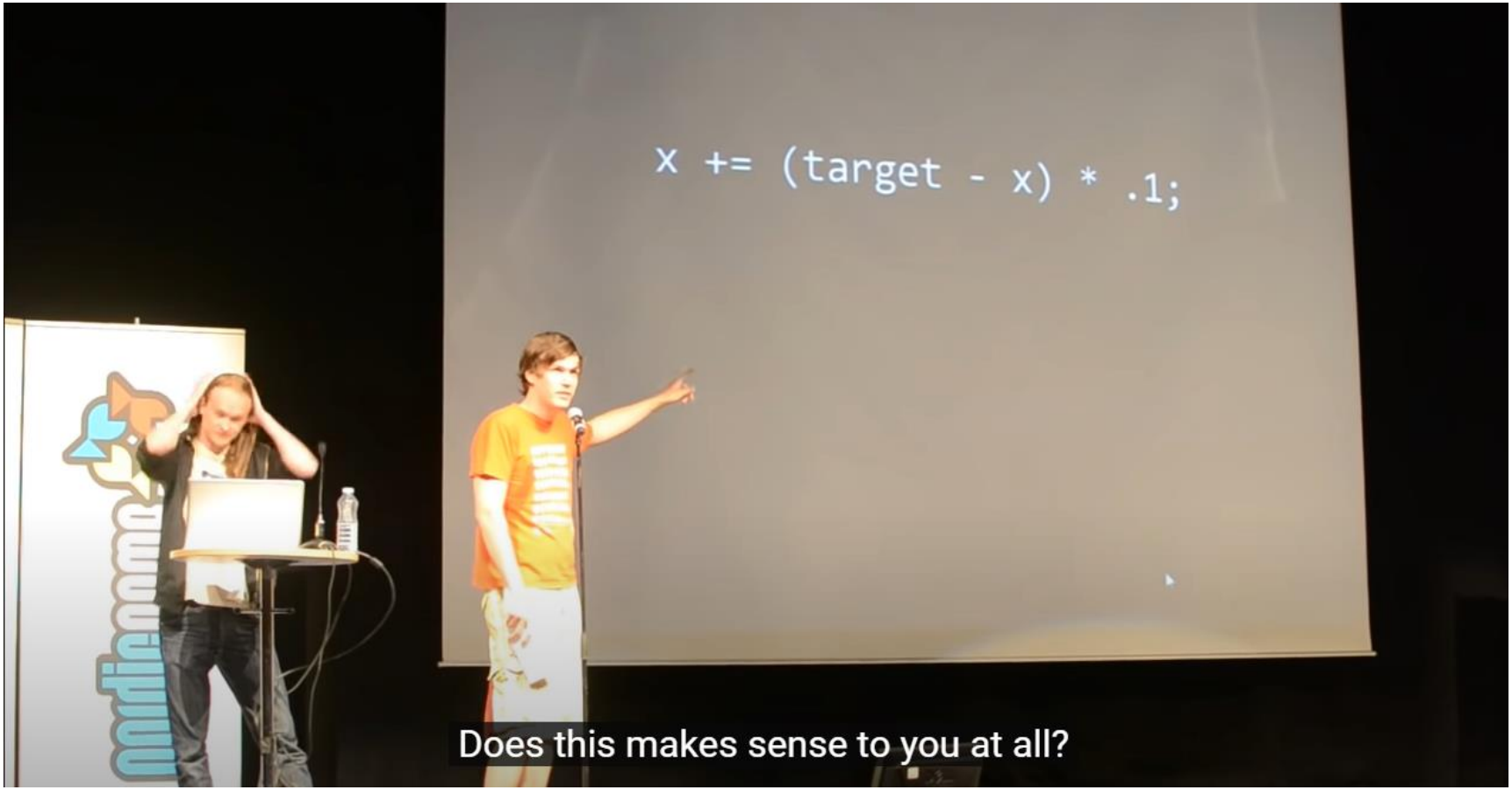
Go and watch…



Juice ==?== Maximum output for minimum (player) input.

Tweening

Go and watch…



The simplest tweening ever, featuring ease-out… you can use it in so many situations. But be sure to include frame time delta in here!

Tweening

Go and watch…

And you can do a lot with tweenings… if applied to right values.

Tweening

Go and watch…

Vlambeer - The art of screenshake [length 44:09]
https://www.youtube.com/watch?v=AJdEqssNZ-U

Tweening

Go and watch…

The talk between the game and the player, discussion even?

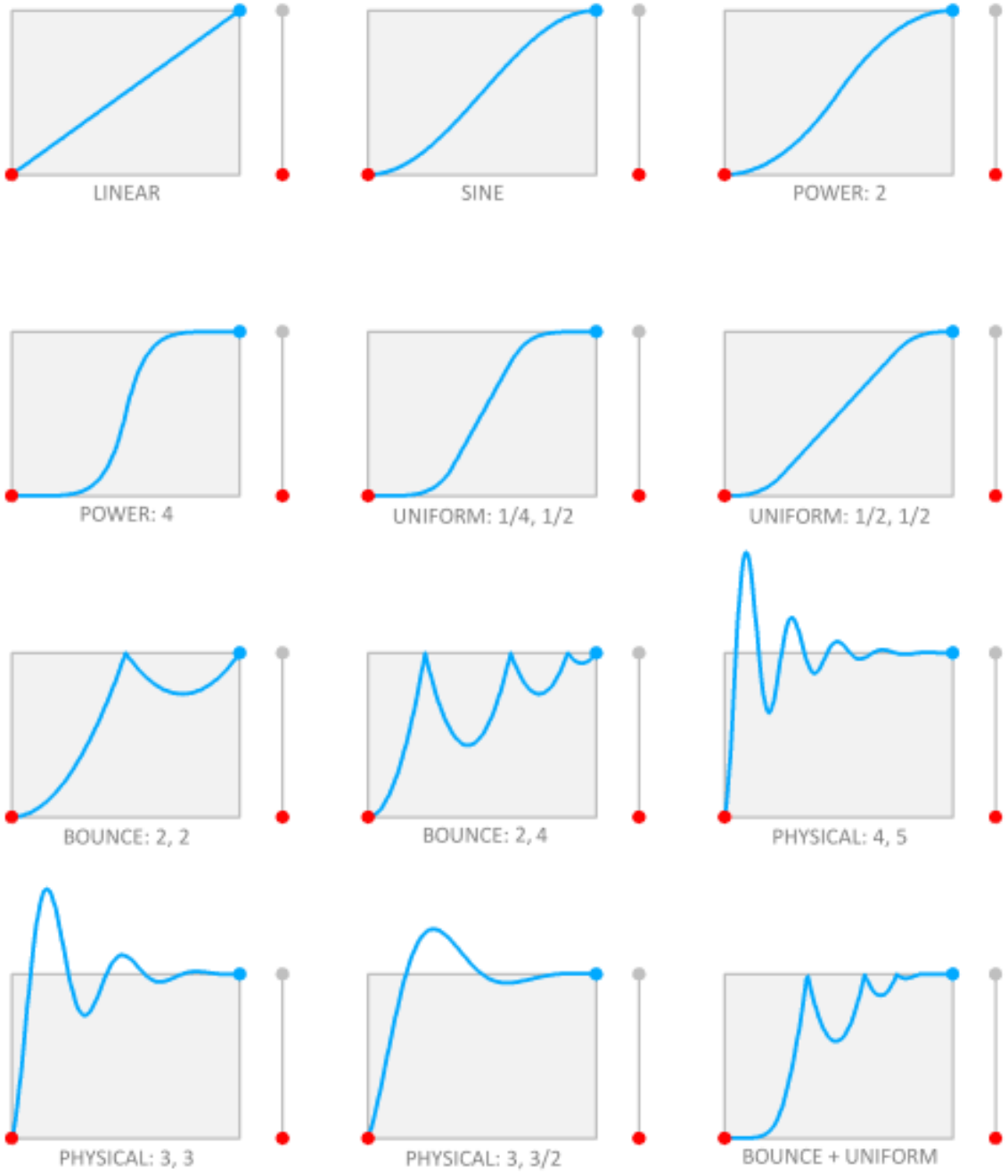
Tweening

Go and watch…

One never has enough tricks in the pocket…

Twee

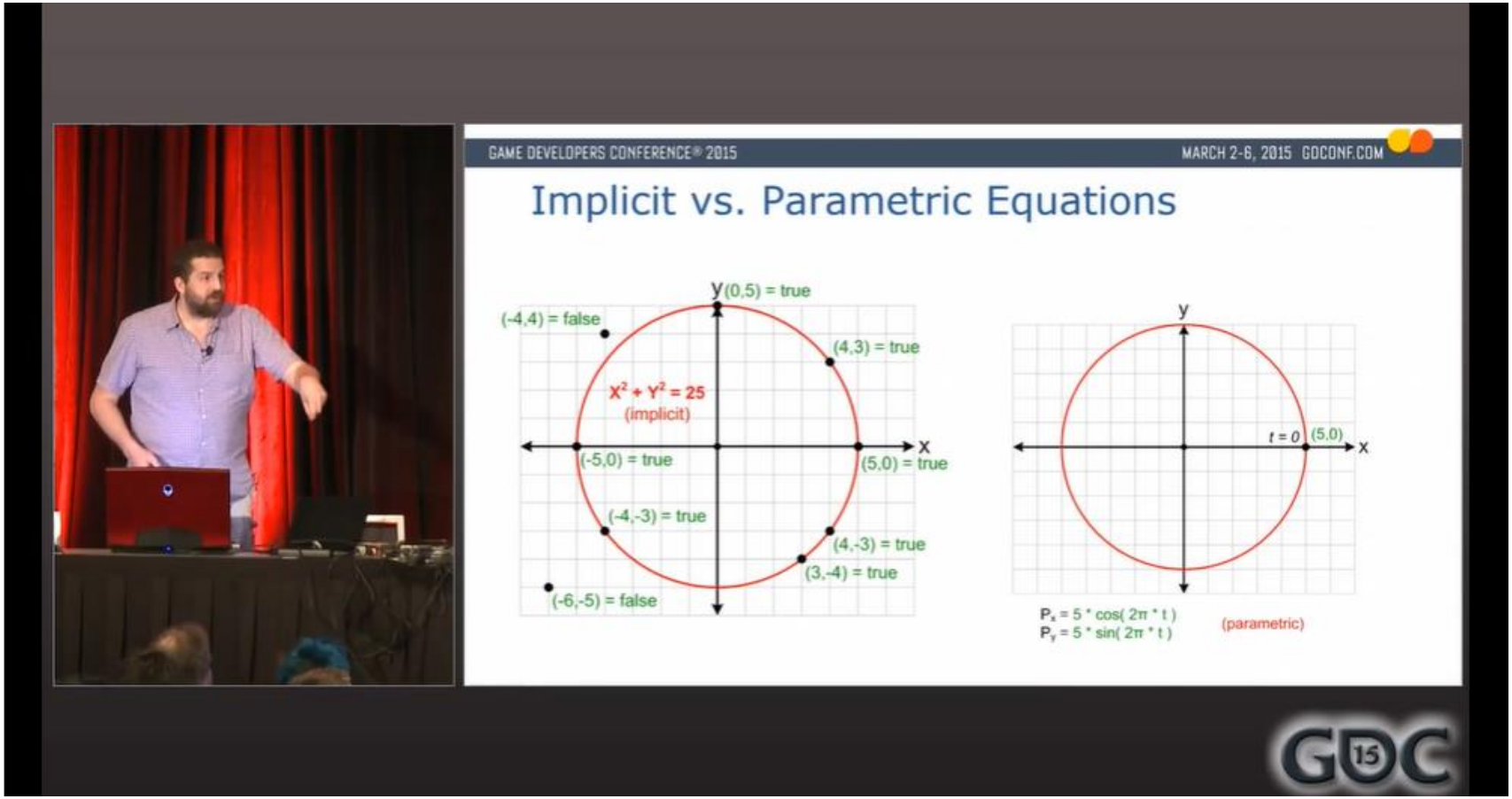
Go anc



1ucasvb.tumblr.com

Tweening

Go and watch…



Math for Game Programmers: Fast and Funky 1D Nonlinear Transformations
https://www.youtube.com/watch?v=mr5xkf6zSzk [length 28:01]

Checklist

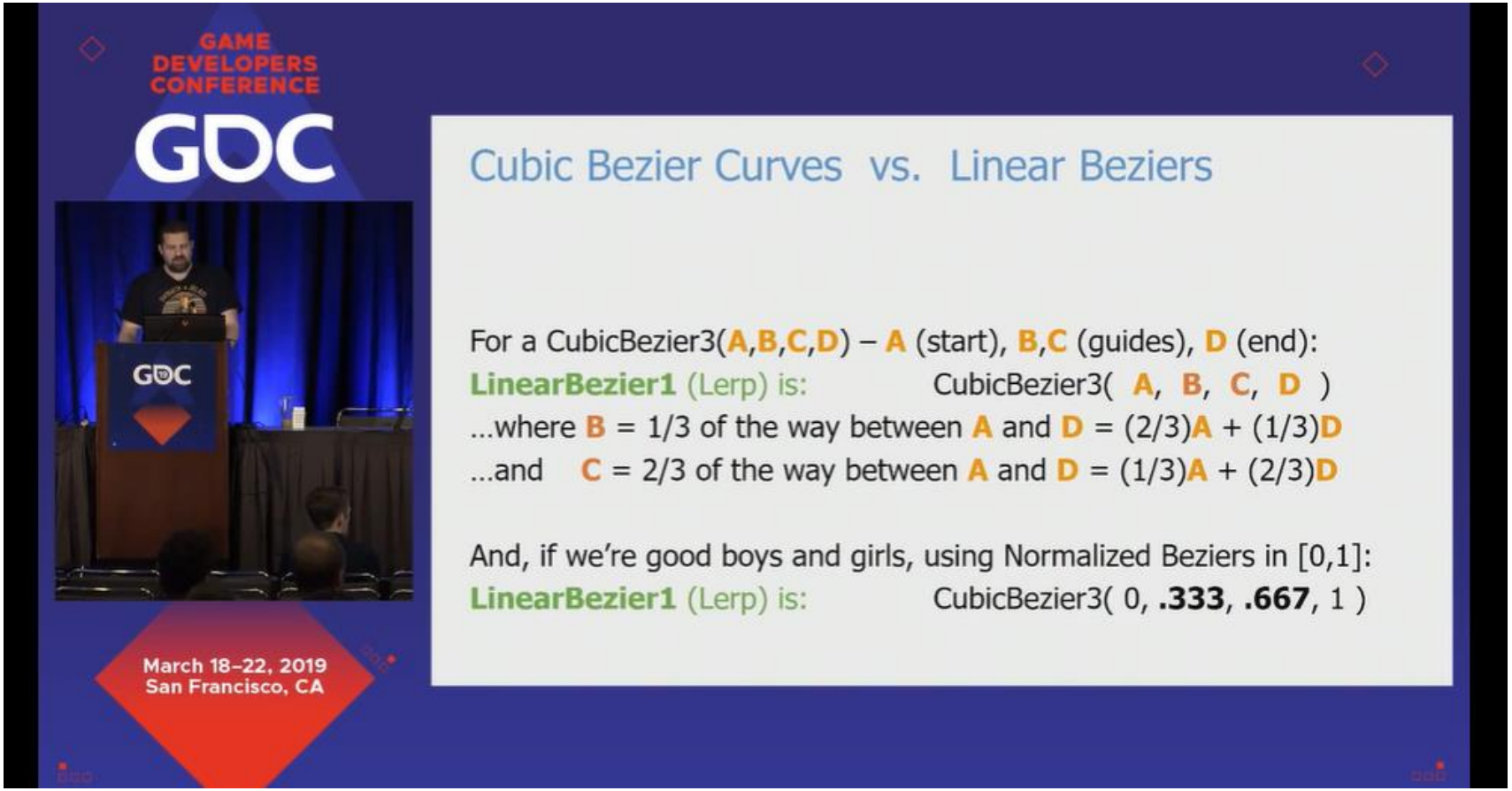
Fast and Funky 1D Nonlinear Transformations

- Implicit vs. Parametric Equations
- Parametric manipulations
  - `tween1(x) = x`
  - `tween2(x) = x^2`
  - `tween1(quad(x)) = x^2 = tween2(x)`
- Number ranges `[0;1]` and `[-1;1]` and why they are important for the (tweening) math
  - Range mapping
- SmoothStart**N**, SmoothStop**N** (and their construction)
- Function mixing (blending), SmoothStart**2.2** without `pow()`
- Funky function construction toolbox

Tweening

Go and watch…

You will need GDC Vault account for this one, DM me…



Math for Game Developers: Curves Revisited [length 1:04:17]
https://www.gdcvault.com/play/1025907/Math-for-Game-Developers-Curves

Checklist

Curves revisited

- Lerp ~ Linear Bezier, Quadratic Bezier, Cubic Bezier
  - De Casteljau's algorithm
  - Non-uniformity in length per step, i.e., “velocity”
- Quadratic Bezier midpoint control
- Easing functions as Quadratic / Cubic Bezier
- Non-invasive curve splitting
- Cubic Hermite curves, explanation of control points
- Splines
  - “velocity” continuity
    - Quadratic vs. Cubic Bezier in spline editing
  - Catmull-Rom

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Material has been produced within and supported by the project „Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“ kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.

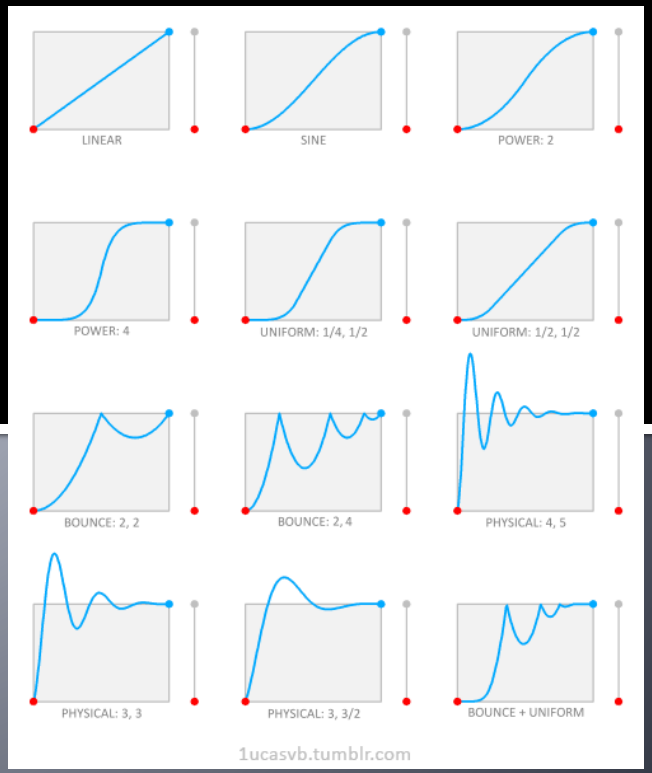
Faculty of Mathematics and Physics
Charles University, Prague
27th November 2023

Gameplay Programming Level 05

(In-Be)Tween(ing)

Part II

Curvy (s)lin(m)es everywhere again

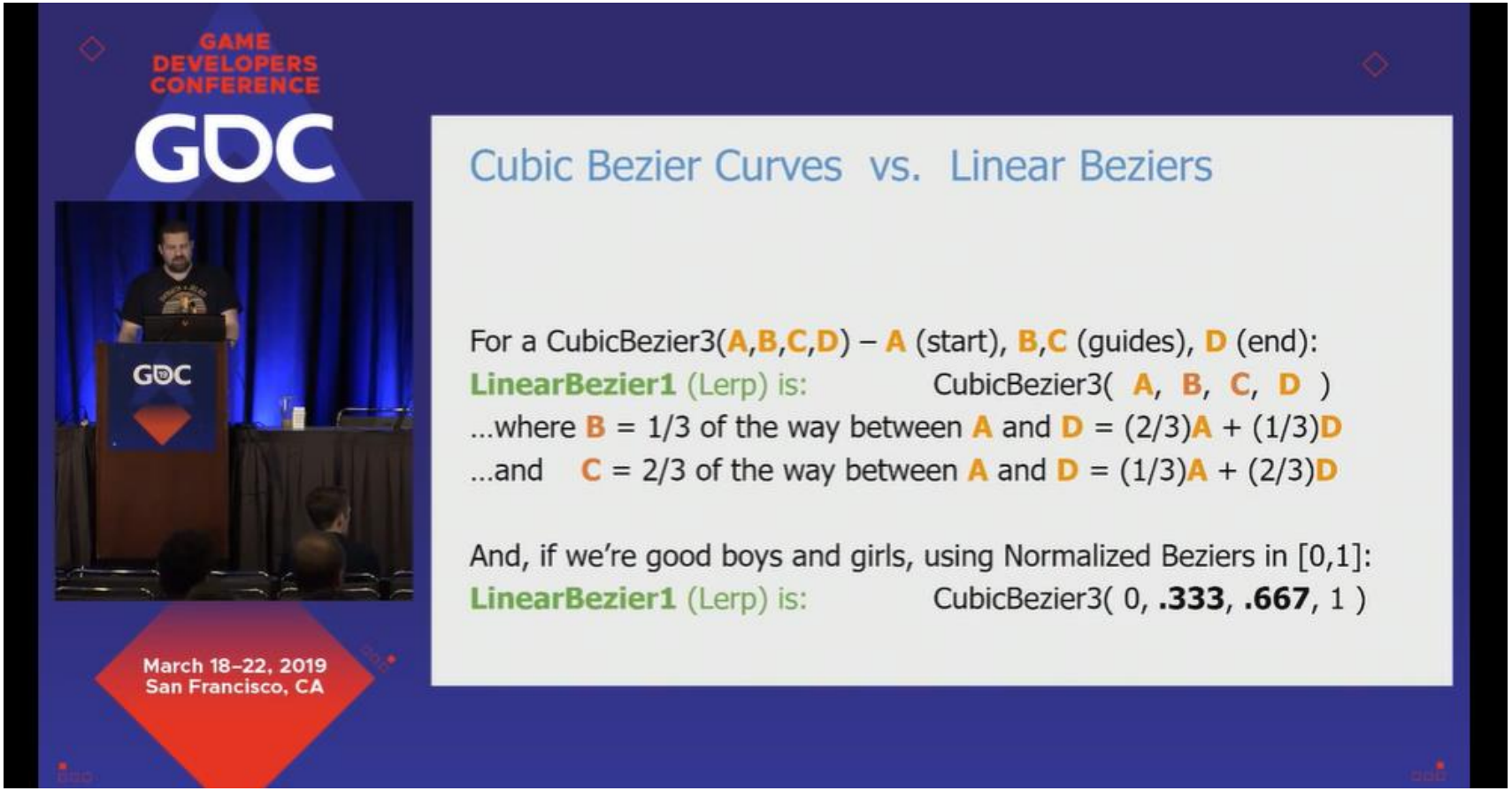


Kudos and credit mainly to:
Squirell Eiserloh

Tweening

You can DM me for access…

We will follow (totally) yet another video…



Math for Game Developers: Curves Revisited [length 1:04:17]

Kudos and credit to: Squirell Eiserloh

https://www.gdcvault.com/play/1025907/Math-for-Game-Developers-Curves

Checklist for today

Curves revisited

- Tweens recap
- Interpolation Lerp ~ Linear Bezier, Quadratic Bezier, Cubic Bezier
  - De Casteljau's algorithm
  - Non-uniformity in length per step, i.e., “velocity”
- Quadratic Bezier midpoint control
- Easing functions as Quadratic / Cubic Bezier
- Non-invasive curve splitting
- Cubic Hermite curves, explanation of control points
- Splines
  - “velocity” continuity
    - Quadratic vs. Cubic Bezier in spline editing
  - Catmull-Rom

Recap

Topic 1

First tweens are...

Tweens

Topic 1

Parametrics

Anything that you plug a number 't' into, and get something out.
Usually: t=0.0 at the "start", and t=1.0 at the "end"

For most of today's talk:

```
Pos(t) = Vec2( somethingWithT, somethingElseWithT );
```

For example:

```
Pos(t) = Vec2( t, 0 )      // point moving east from x=0 to x=1
Pos(t) = Vec2( t, t )      // point moves diagonally north-east
Pos(t) = Vec2( t, sin(t) ) // bobs up/down as it moves east
Pos(t) = Vec2( cos(t), sin(t) ) // moving around in a circle
```

Tweens

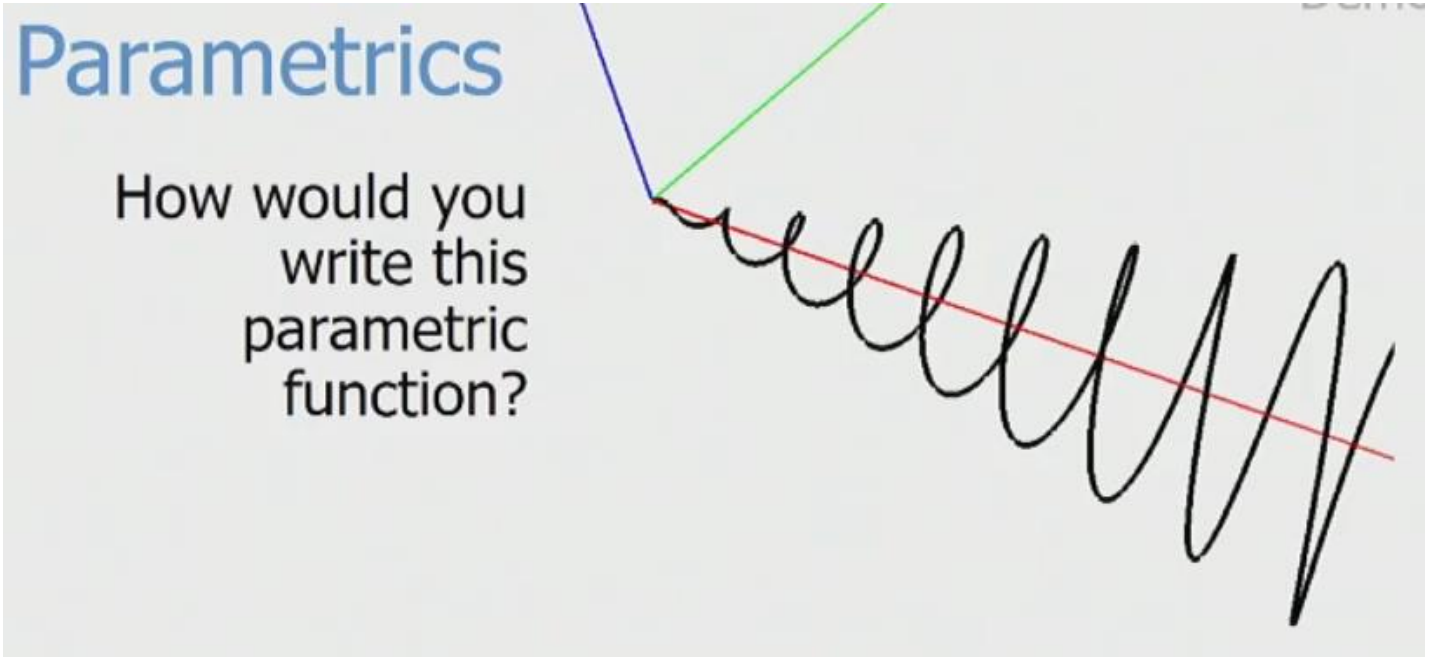
Topic 1

Parametrics

Implicit Equations	vs.	Parametric Equations
$y = x$		$P(t) = (t , t)$
(x,y) is on the **line** if true		Gets an (x,y) for any t (P moves in a **line**)

Tweens

Topic 1

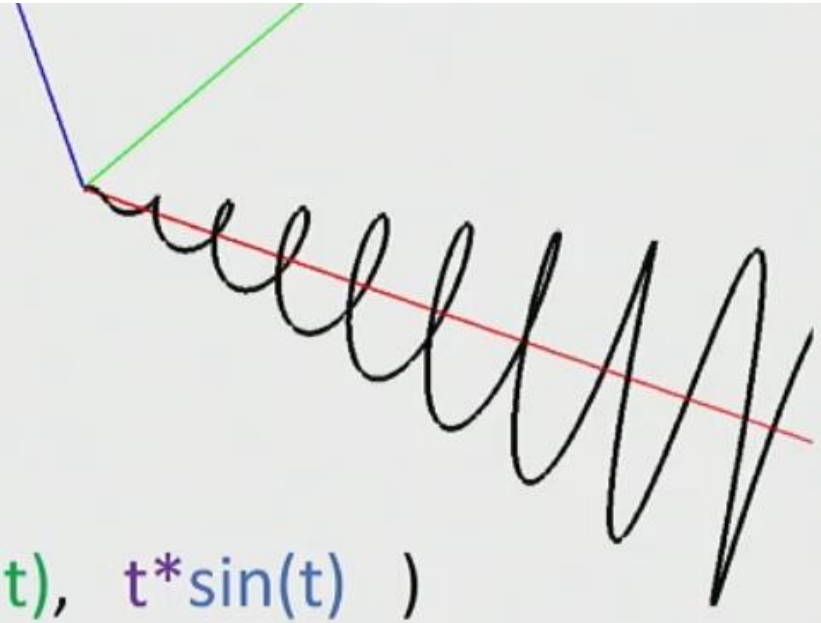


Tweens

Topic 1

Parametrics

How would you write this parametric function?



$P(t) = (\ t,\ t*\cos(t),\ t*\sin(t)\ )$

- As t increases, move along the +x axis at a fixed rate
- …while circling the x-axis with (cos,sin) in y,z
- …while expanding the circle's radius outward

Tweens

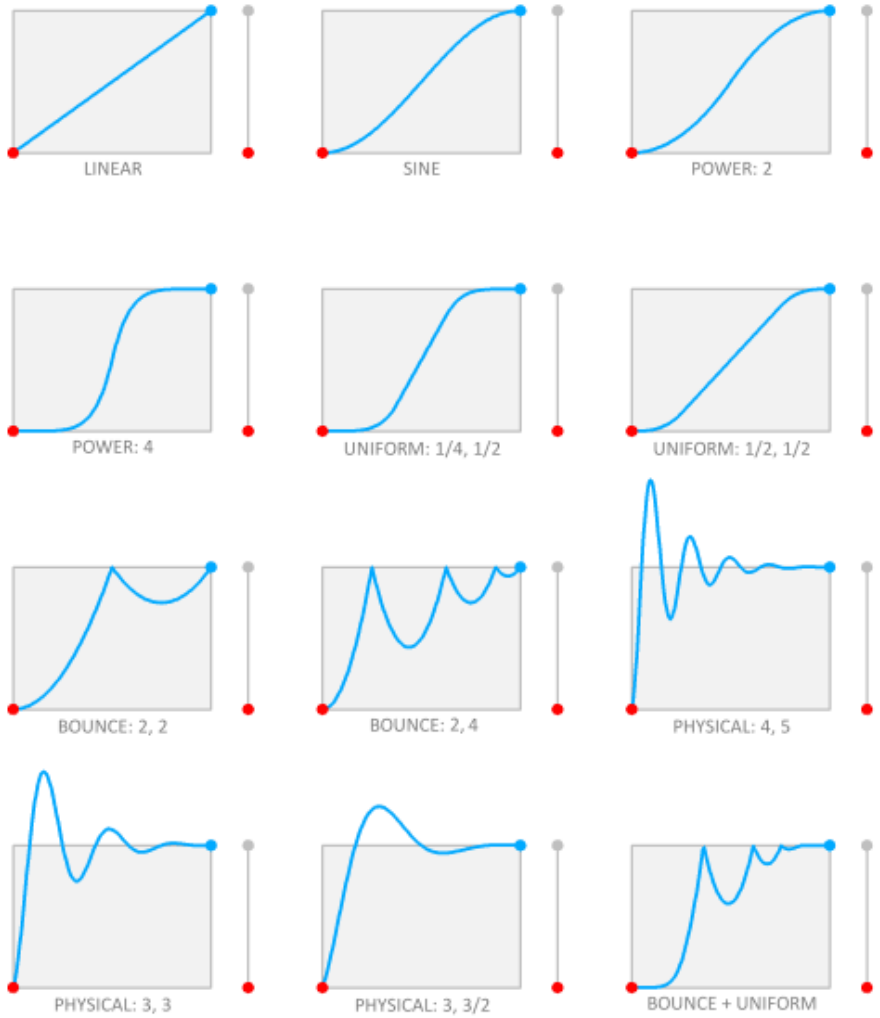
Topic 1

SmoothStartN(x) = x^N

SmoothStopN(x) = $1-(1-x)^N$

Parametric manipulations

- `tween1(x) = x`
- `tween2(x) = x^2`
- `tween1(quad(x)) = x^2 = tween2(x)`



1ucasvb.tumblr.com

Interpolation

Topic 2

How to move between two numbers

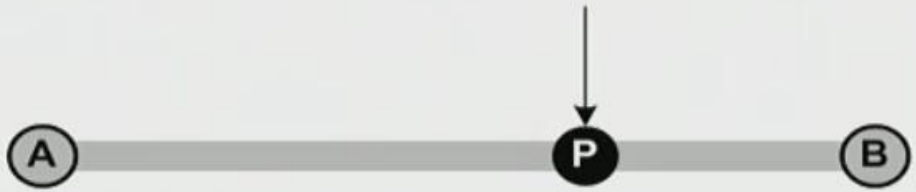
Interpolation

Topic 2

Interpolation – Averaging and Blending

We can also blend A and B unevenly (with different **weights**):

$$(.35 * A) + (.65 * B)$$



In this case, we are blending "35% of A with 65% of B".

Use any blend weights we want, as long as they add up to 1.0 (100%)

Interpolation

Topic 2

Interpolation – Averaging and Blending

So if we generalized it, we would say:

(**s** * **A**) + (**t** * **B**)

...where **s** is "how much of **A**" we want,
and **t** is "how much of **B**" we want

...and **s** + **t** = 1.0 (really, **s** is just 1-**t**)

so: ((1-**t**) * **A**) + (**t** * **B**)

Which means we can control the balance of the entire blend by changing just one number: **t**

Let's call this function "**Lerp**" (for **L**inear Int**erp**olation)

Interpolation

Topic 2

Interpolation

Let's call this function "**Lerp**" (for **L**inear Int**erp**olation), as:

Lerp(**A**, **B**, **t**) = ?

There are three ways to think about this,
and three ways to write it:

#1: (**s** * **A**) + (**t** * **B**) Think: "A 60/40 blend of B and A"

#2: ((1-**t**) * **A**) + (**t** * **B**) Think: "60% of B, and A gets what's left"

#3: **A** + **t***(**B** – **A**) Think: "Start at A, then go 60% of the journey from A to B"

Interpolation

Topic 2

Interpolation – Two Types

Linear Interpolation

(a.k.a. Lerp)

and

Nonlinear Interpolation

(curves, splines, easing functions, etc.)

Interpolation

Topic 2

Nonlinear Interpolation – Easing

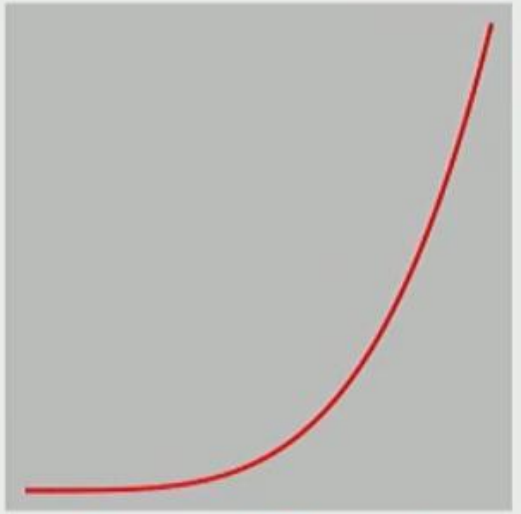
Demo: Easing

Easing / Tweening functions

We still want to go from **A** to **B**, but not in a boring linear way...

Examples:

SmoothStart2 (a.k.a. Quadratic Ease-In)
SmoothStart3 (a.k.a. Cubic Ease-In)
SmoothStart4 (a.k.a. Quartic Ease-In)
SmoothStop2 (a.k.a. Quadratic Ease-Out)
SmoothStop3 (a.k.a. Cubic Ease-Out)
SmoothStop4 (a.k.a. Quartic Ease-Out)



SmoothStart4

Interpolation

Topic 2

Nonlinear Interpolation – Easing

Easing / Tweening functions

We still want to go from **A** to **B**, but not in a boring linear way...

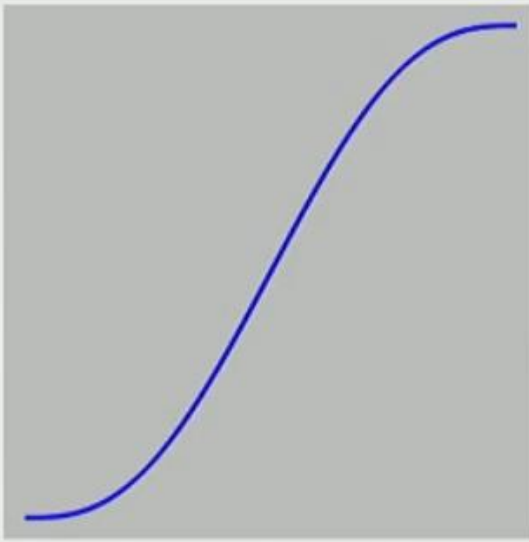
Examples:

SmoothStep3 (a.k.a. Cubic Ease-In-Ease-Out)

SmoothStep5 (a.k.a. Quintic Ease-In-Ease-Out)

Wait, what happened to SmoothStep4??

? ? ? ? ?



SmoothStep5

Interpolation

Topic 2

Nonlinear Interpolation – Easing

Easing / Tweening functions

We still want to go from **A** to **B**, but not in a boring linear way...

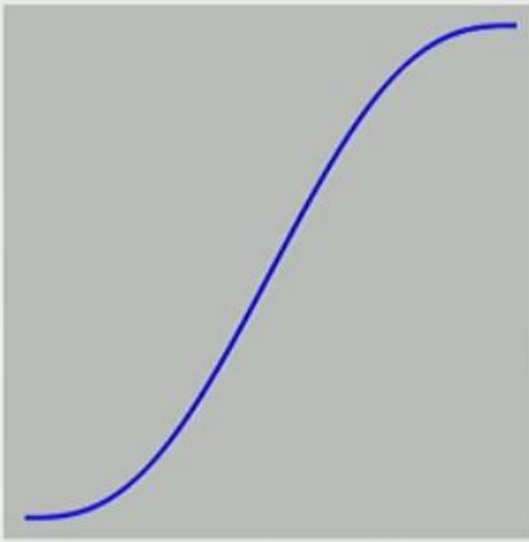
Examples:

SmoothStep3 (a.k.a. Cubic Ease-In-Ease-Out)

SmoothStep5 (a.k.a. Quintic Ease-In-Ease-Out)

Wait, what happened to SmoothStep4??

? ? ? ? ?



SmoothStep5

Interpolation

Topic 2

Normalized Nonlinear Interpolation – "Pure" Easing

"Pure" Easing / Tweening functions

Instead of going from **A** to **B**, we simplify things by going from **0** to **1** instead – but still in a completely non-boring way!

Non-normalized Nonlinear Interpolation

SmoothStart3 (Cubic Ease-In): $F(t) = s + (e-s)[\ (t-s)/(e-s)\ ]^3$

Normalized Nonlinear Interpolation

SmoothStart3 (Cubic Ease-In): $F(t) = t^3$

Interpolation

Topic 2

Normalized Nonlinear Interpolation – "Pure" Easing

Squirrel's Principles of Parametrics™

1. It's almost always easier to modify **t**, rather than **F**(t)

2. Life is better in [0,1]! (loads of math disappears!)

Therefore:

Normalized Nonlinear Interpolations **for the win!**

Interpolation

Topic 3

(Quadratic) Bezier Curves

a.k.a.

Lerp-of-lerps

Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.1

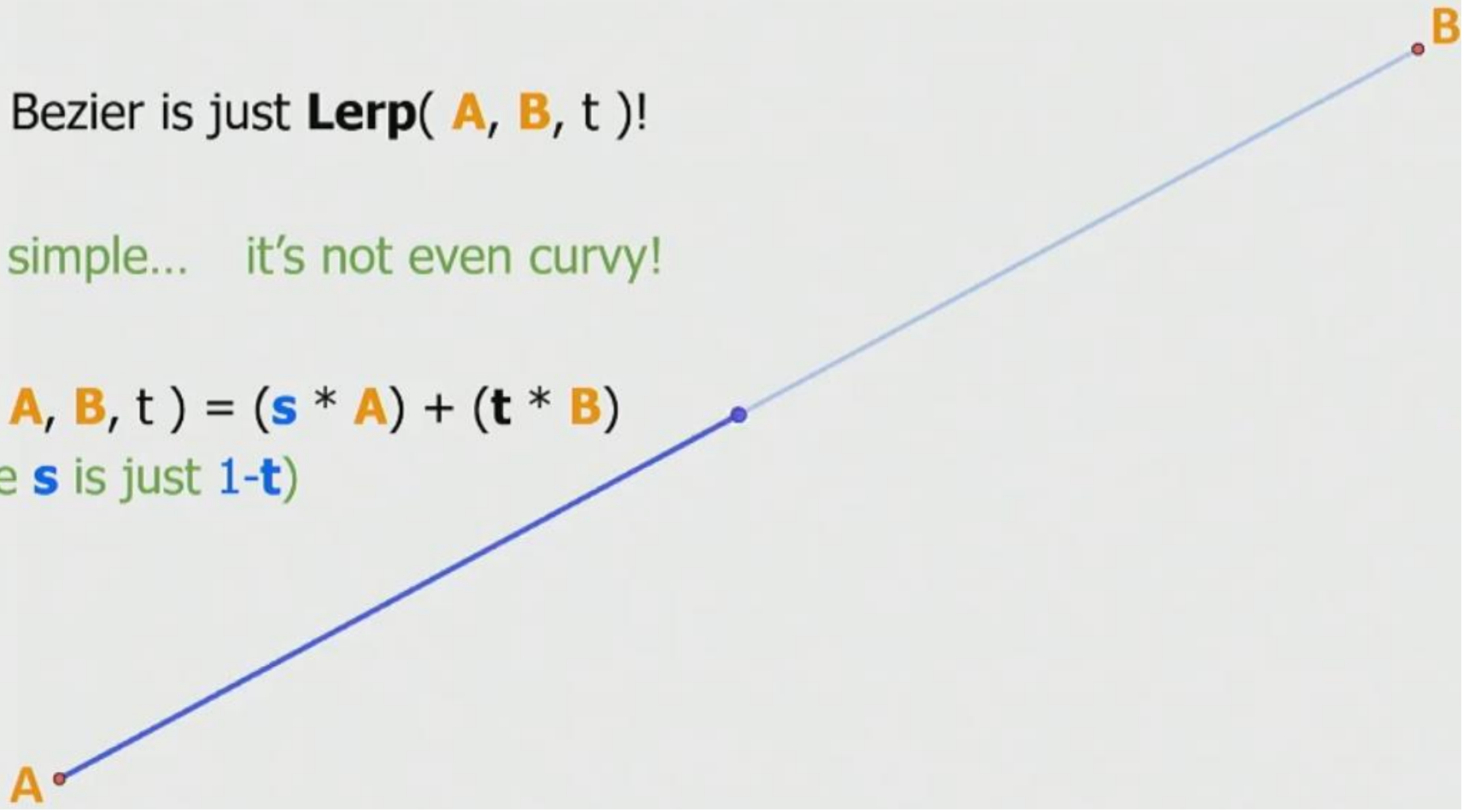
Linear Bezier Curves

Linear Bezier is just **Lerp**(**A**, **B**, t)!

It's so simple… it's not even curvy!

Lerp(**A**, **B**, t) = (**s** * **A**) + (**t** * **B**)

(where **s** is just 1-**t**)

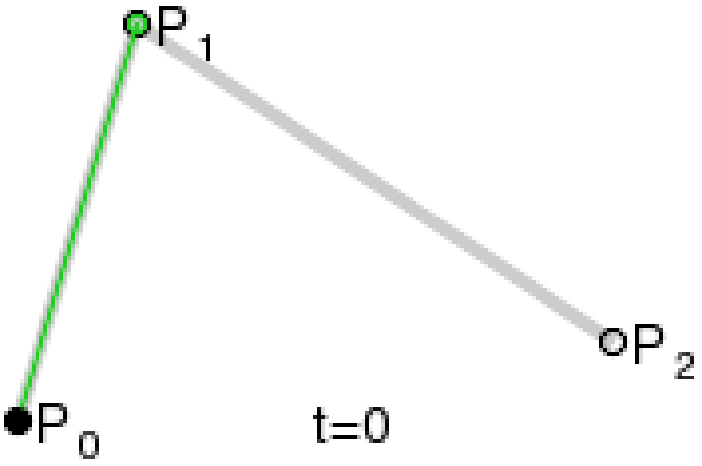


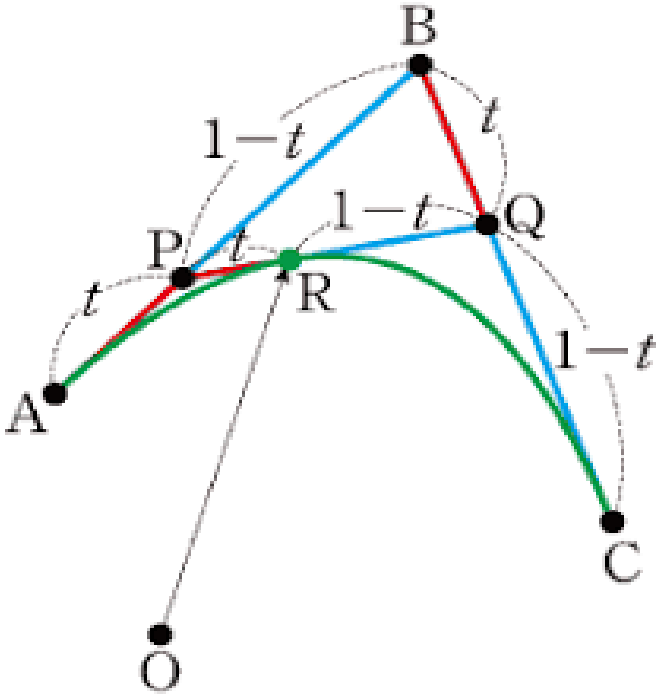
Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.1

Quadratic Bezier Curve

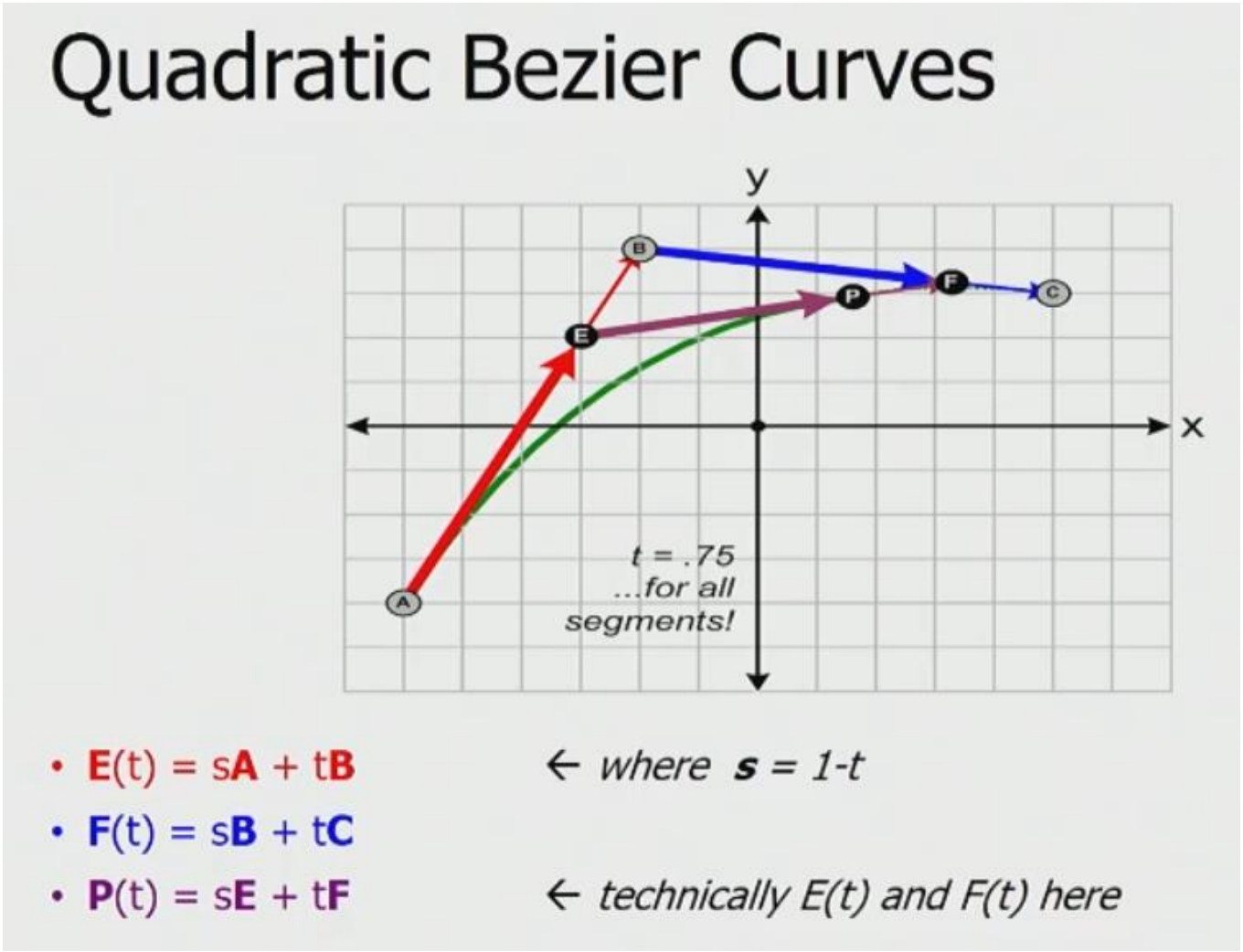
De Casteljau's algorithm – Lerp of lerps





Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.1



Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.1

Quadratic Bezier Curves

» One equation to rule them all:

$$E(t) = sA + tB$$
$$F(t) = sB + tC$$
$$P(t) = sE(t) + tF(t)$$

or

$$P(t) = s(\ sA + tB\ ) + t(\ sB + tC\ )$$

or

$$P(t) = (s^2)A + (st)B + (st)B + (t^2)C$$

or

$$P(t) = (s^2)A + 2(st)B + (t^2)C$$

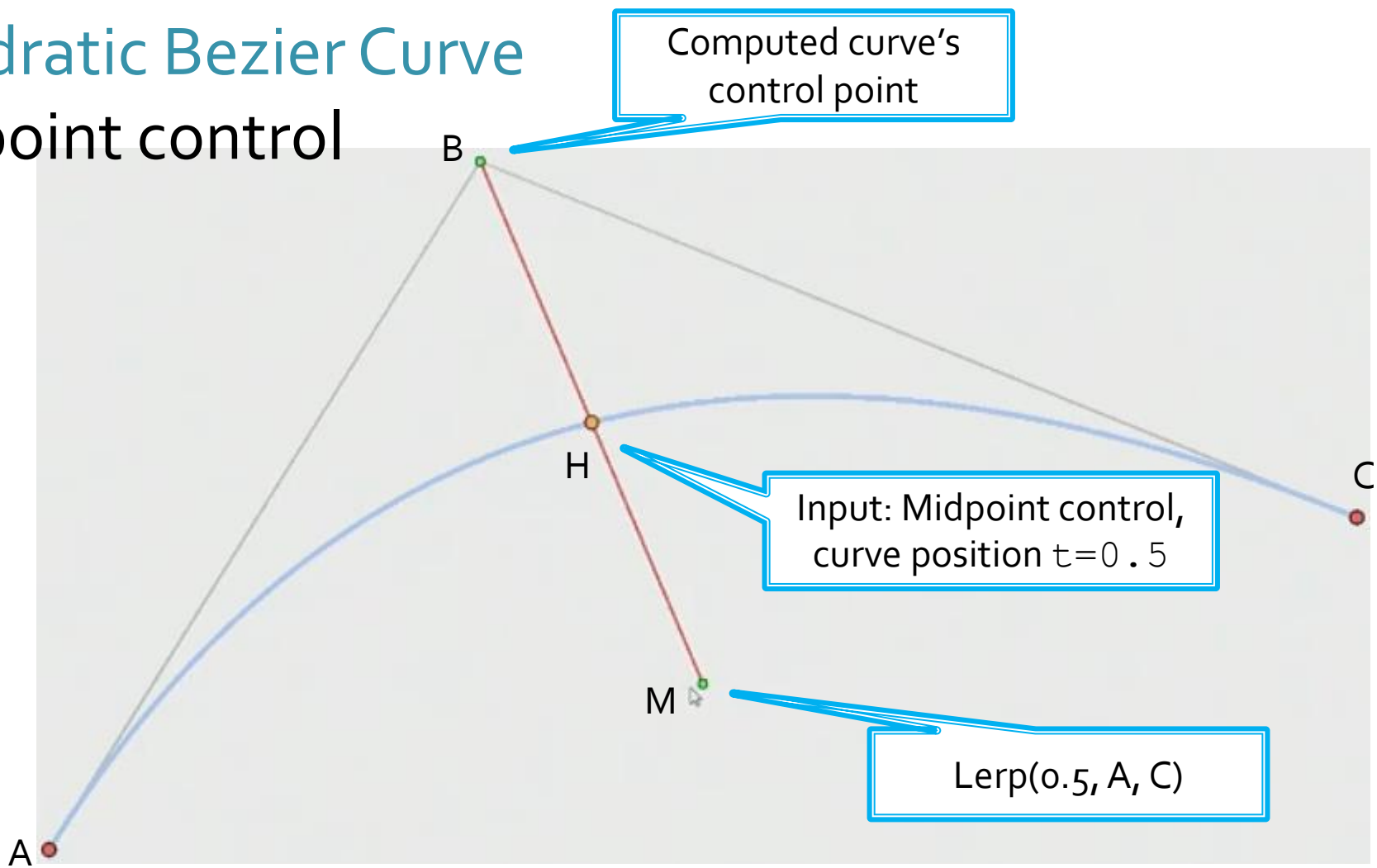
(BTW, there's our "quadratic" t^2)

Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.2

Quadratic Bezier Curve

Midpoint control



Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.2

Quadratic Bezier Curves – Midparam Control

Quadratic Beziers can also be controlled by a "midparam" method:

1. The user simply picks and drags a parametric halfway point **H** (i.e. the point at which they want the curve to be at t=0.5).

2. The code finds the midpoint between start (**A**) and end (**C**).

3. The code finds displacement from midpoint **M** to halfway point **H**

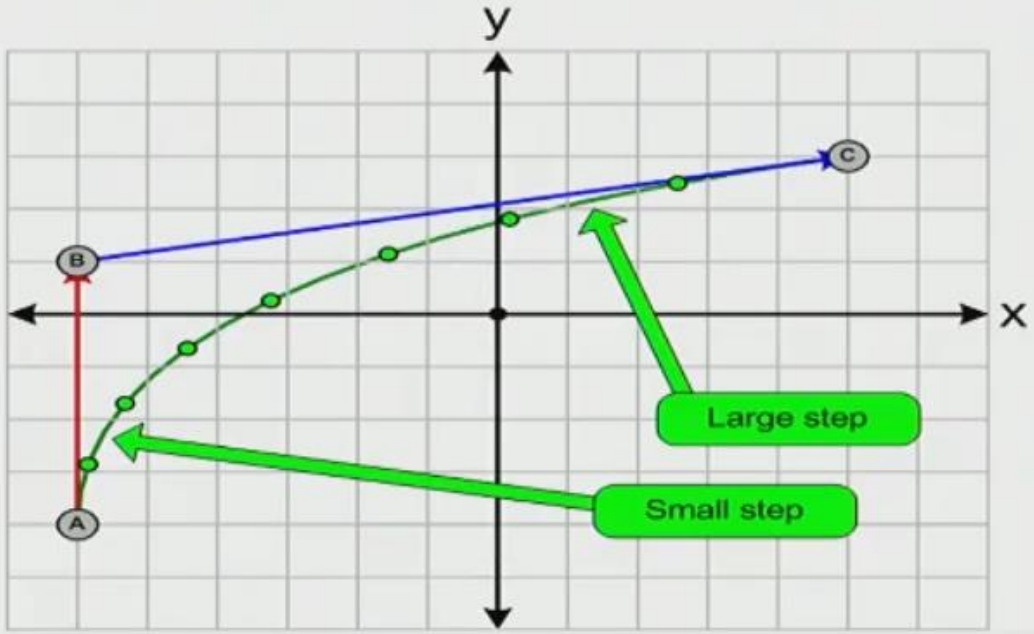
4. The code deduces the location of (**B**) to be **B** = **H** + (**H**–**M**)

Bezier Curves a.k.a. lerp(s)-of-lerps

Topic 3.3

Non-uniformity

- Be careful: most curves are not **uniform**; that is, they have variable "density" or "speed" throughout them.



Cubic Bezier Curves

Topic 4

Lerp of two quadratic B. c.

~

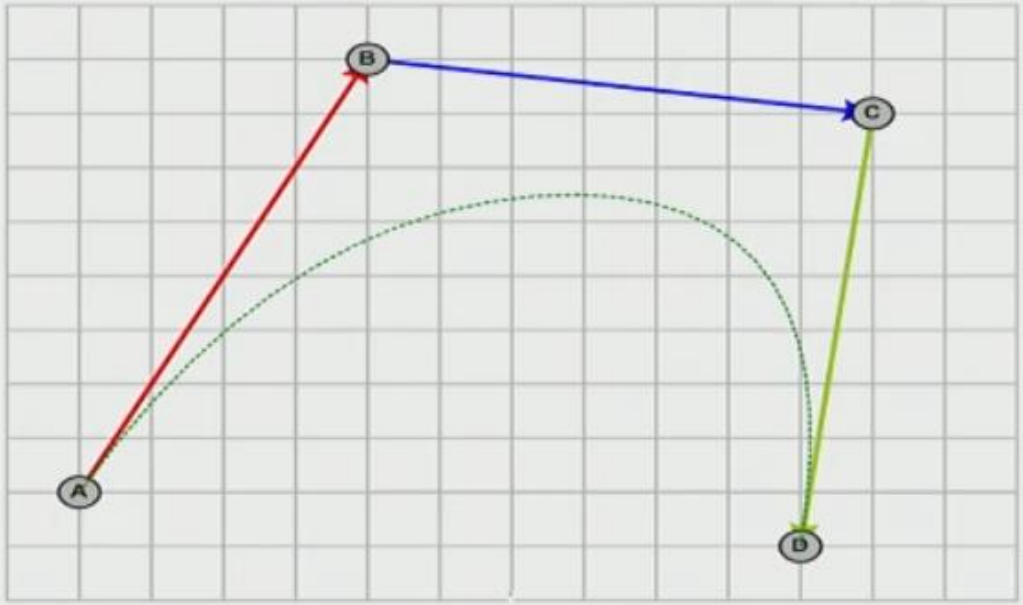
Lerp of two lerp-of-lerps

Cubic Bezier Curves

Topic 4

~25:10

Cubic Bezier Curves

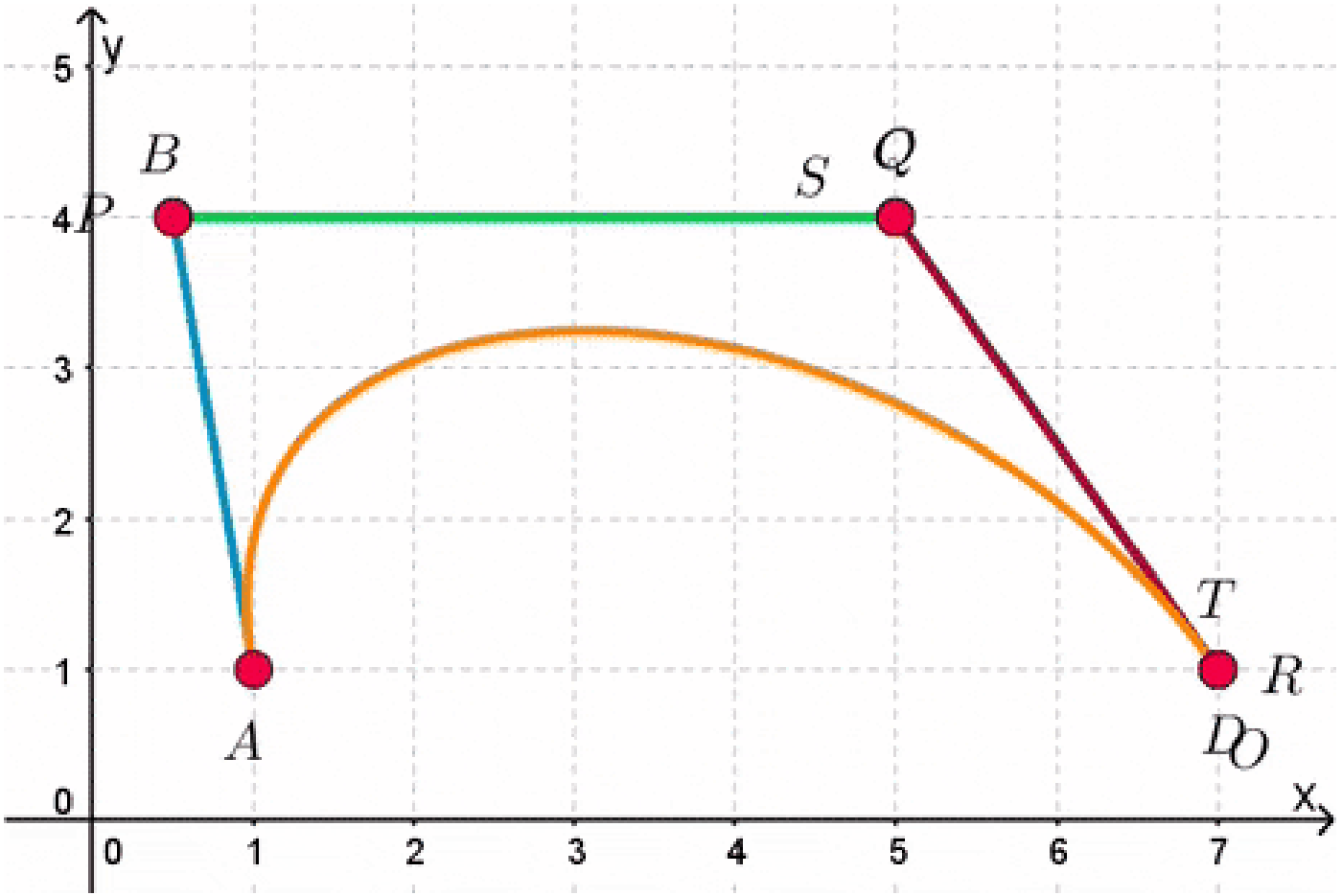


- Four **control points**: A, B, C, and D
- 3 different Linear Beziers: AB, BC, and CD
- 2 different Quadratic Beziers: ABC and BCD

Cubic Bezier Curves

Topic 4

~25:10



Cubic Bezier Curves

Topic 4

~25:10

Cubic Bezier Curves

- $E(t) = sA + tB$ // Linear Bezier (Lerp of A and B)
- $F(t) = sB + tC$ // Linear Bezier (Lerp of B and C)
- $G(t) = sC + tD$ // Linear Bezier (Lerp of C and D)

- $Q(t) = sE + tF$ // Quadratic Bezier (Lerp of E and F)
- $R(t) = sF + tG$ // Quadratic Bezier (Lerp of F and G)

- $P(t) = sQ + tR$ // Cubic Bezier (Lerp of Q and R)

- Okay! So let's combine these all together...

Cubic Bezier Curves

Topic 4

~25:10

Cubic Bezier Curves

- Do some hand-waving mathemagic substitutions…
  …and we get **one equation to rule them all**:

$$\mathbf{P(t)} = (s^3)\mathbf{A} + 3(s^2t)\mathbf{B} + 3(st^2)\mathbf{C} + (t^3)\mathbf{D}$$

(BTW, there's our "cubic" $\mathbf{t^3}$)

Cubic Bezier Curves

Topic 4

~25:10

Cubic Bezier Curves

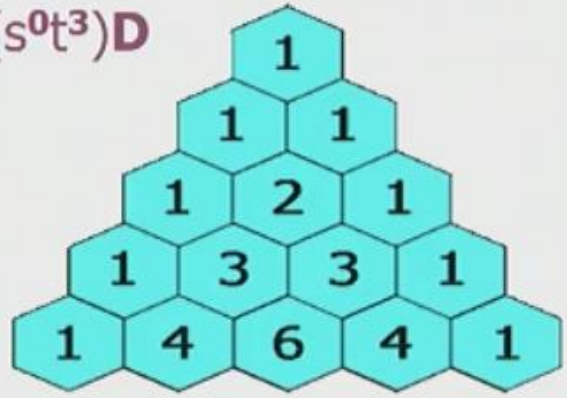
- Write in all of the numeric coefficients...
- Express each term as powers of **s** and **t**

$$P(t) = \mathbf{1}(s^1t^0)\mathbf{A} + \mathbf{1}(s^0t^1)\mathbf{B}$$

$$P(t) = \mathbf{1}(s^2t^0)\mathbf{A} + \mathbf{2}(s^1t^1)\mathbf{B} + \mathbf{1}(s^0t^2)\mathbf{C}$$

$$P(t) = \mathbf{1}(s^3t^0)\mathbf{A} + \mathbf{3}(s^2t^1)\mathbf{B} + \mathbf{3}(s^1t^2)\mathbf{C} + \mathbf{1}(s^0t^3)\mathbf{D}$$

- Note: numeric coefficients...
  are from Pascal's Triangle



Quad. Bez. Cur. And Easing Funcs

Topic 5

On the relations of…

Quad. Bez. Cur. And Easing Funcs

Topic 5

~25:10

Quadratic Bezier Curves vs. Easing Functions

In 2015, we showed different ways to mathematically, geometrically, and intuitively derive common and custom Easing functions, such as SmoothStop2 / Ease-Out Quadratic.

With Bezier curves, **there's an even easier way**!

For a QuadraticBezier2(**A**,**B**,**C**) – **A** (start), **B** (guide), and **C** (end):

SmoothStart2 (Ease-In Quadratic) is: QuadraticBezier2(**A**,**A**,**C**)

SmoothStop2 (Ease-Out Quadratic) is: QuadraticBezier2(**A**,**C**,**C**)

And, if we're good boys and girls, using Normalized Beziers in [0,1]:

SmoothStart2 (Ease-In Quadratic) is: QuadraticBezier2(0,**0**,1)

SmoothStop2 (Ease-Out Quadratic) is: QuadraticBezier2(0,**1**,1)

Quad. Bez. Cur. And Easing Funcs

Topic 5

~32:20

Cubic Bezier Curves vs. Easing Functions

In 2015, we derived Easing functions SmoothStart3, SmoothStop3.

With Cubic Bezier curves, **there's an even easier way**!

For a CubicBezier3(**A**,**B**,**C**,**D**) – **A** (start), **B**,**C** (guides), **D** (end):

SmoothStart3 (Ease-In Cubic) is:[1] CubicBezier3(**A**,**A**,**A**,**C**)

SmoothStop3 (Ease-Out Cubic) is: CubicBezier3(**A**,**C**,**C**,**C**)

And, if we're good boys and girls, using Normalized Beziers in [0,1]:

SmoothStart3 (Ease-In Cubic) is: CubicBezier3(0,**0**,**0**,1)

SmoothStop3 (Ease-Out Cubic) is: CubicBezier3(0,**1**,**1**,1)

Quad. Bez. Cur. And Easing Funcs

Topic 5

~36:00

Cubic Bezier Curves vs. Easing Functions

We also derived SmoothStep3, and Hesitate3, as:

SmoothStep3 = **Lerp**(SmoothStart2, SmoothStop2, **t**)

Hesitate3 = **Lerp**(SmoothStop2, SmoothStart2, **t**)

With Cubic Bezier curves, **there's an even easier way**!

For a CubicBezier3(**A**,**B**,**C**,**D**) – **A** (start), **B**,**C** (guides), **D** (end):

SmoothStep3 (Ease-In-Ease-Out Cubic) is: CubicBezier3(**A**,**A**,**D**,**D**)

Hesitate3 is: CubicBezier3(**A**,**D**,**A**,**D**)

And, if we're good boys and girls, using Normalized Beziers in [0,1]:

SmoothStep3 (Ease-In-Ease-Out Cubic) is: CubicBezier3(0,**0**,**1**,1)

Hesitate3 is: CubicBezier3(0,**1**,**0**,1)

Non-invasive Curve Splitting

Topic 6

Split curves into multiple curves

… camera paths

… enemy paths

…

Non-invasive Curve Splitting

Topic 6

~36:00

Non-Invasive Curve Splitting – Quadratic Bezier

Step 1: Select (and evaluate P at) the parametric split point.

Step 2: Identify the A,B,C nodes of the first sub-curve.

These are:

Split1.A = Parent.A

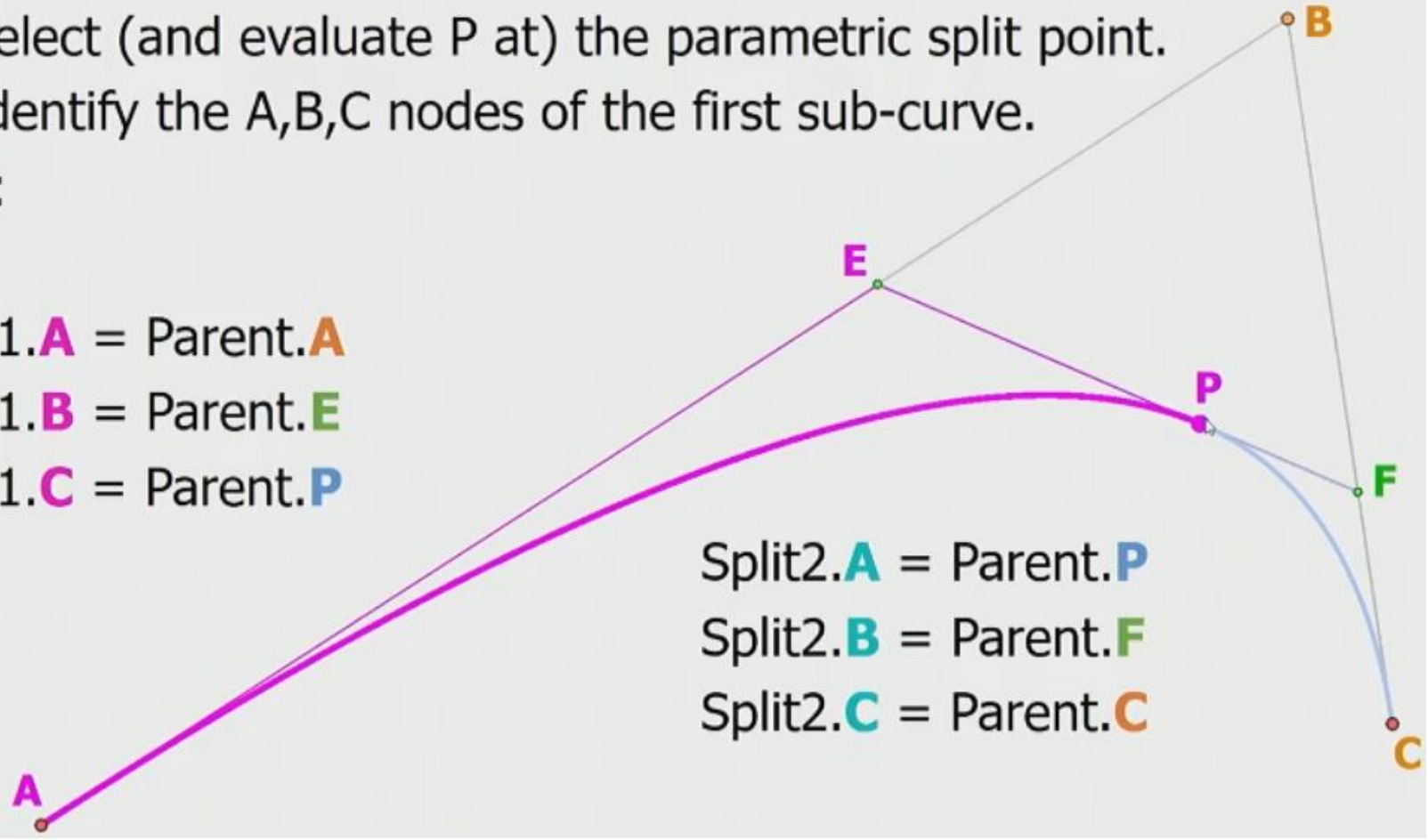
Split1.B = Parent.E

Split1.C = Parent.P

Split2.A = Parent.P

Split2.B = Parent.F

Split2.C = Parent.C



Cubic Hermit Cubes

Topic 7

Sort of different C.B.C.

Easier to work with in practice

Cubic Hermit Cubes

Topic 7

~42:50

Cubic Hermite Curves

- Hermite curves, and Hermite splines, are also parametric and work basically the same way as Bezier curves: plug in "t" and go!

- The formula for **cubic Hermite curve** is:

$$\mathbf{P(t)} = s^2(1+2t)\mathbf{A} + t^2(1+2s)\mathbf{D} + s^2t\mathbf{U} - st^2\mathbf{V}$$

Note, NOT: $\mathbf{P(t)} = s^2(1+2t)\mathbf{A} + t^2(1+2s)\mathbf{D} + s^2t\mathbf{U} + st^2\mathbf{V}$

Cubic Hermit Cubes

Topic 7

~42:50

Cubic Hermite Curves

NEVER FEAR! Cubic Hermites are Stupid Easy!

Cubic **Hermite and Bezier curves are identical** (isomorphic) and can be freely converted back and forth.

To convert from cubic Hermite to Bezier:

$B = A + (U/3)$

$C = D - (V/3)$

To convert from cubic Bezier to Hermite:

$U = 3(B - A)$

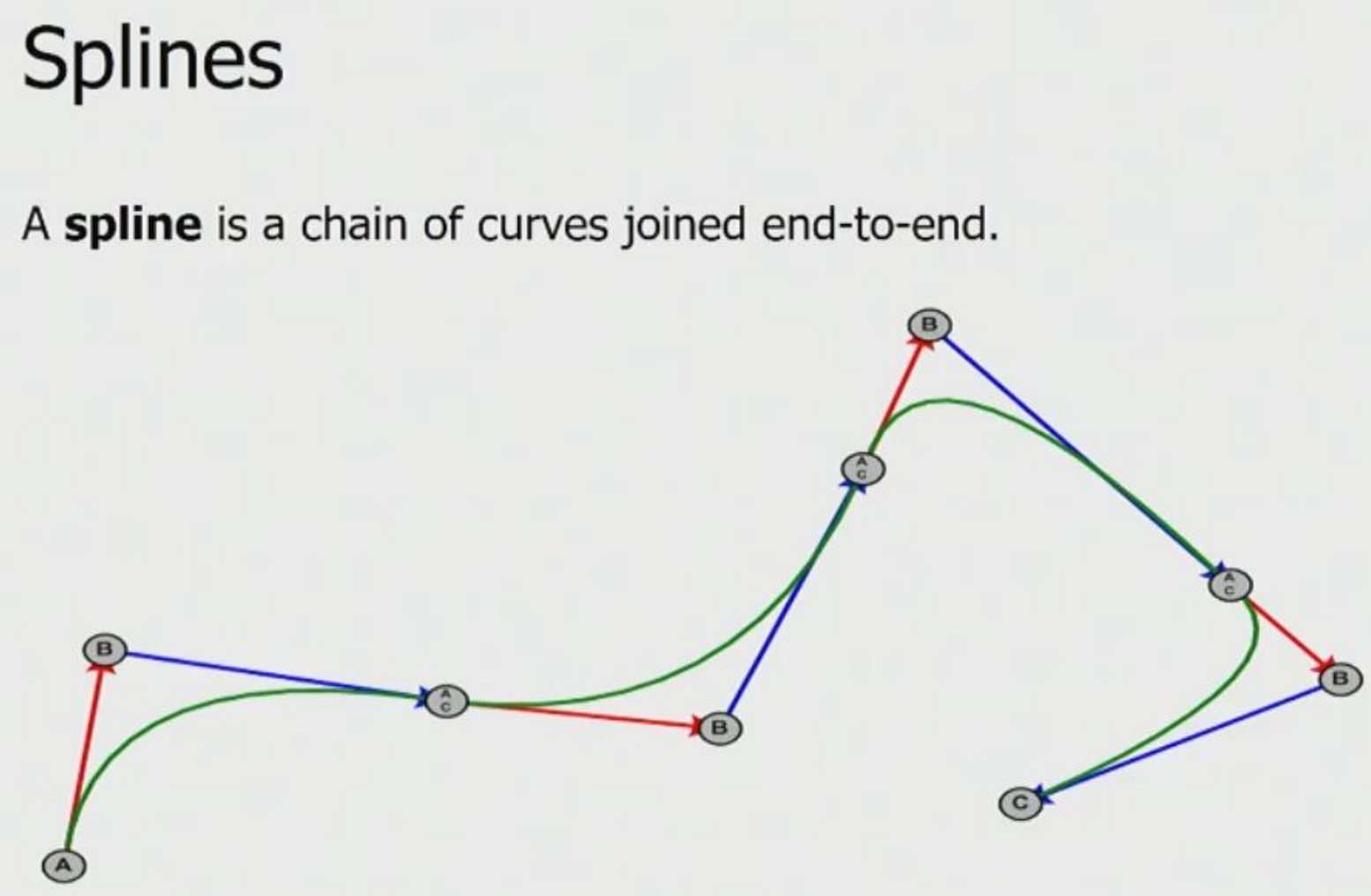
$V = 3(D - C)$

Splines

Topic 8

Chain of curves
connecting on each others

Splines
Topic 8

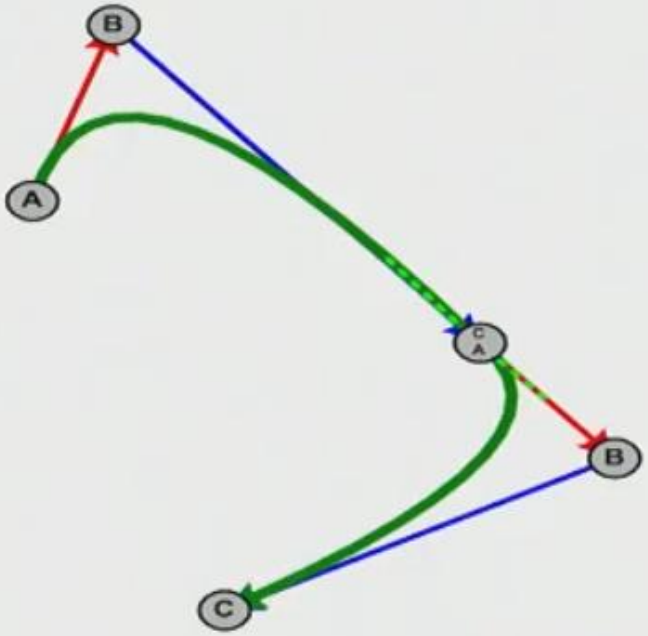


Splines

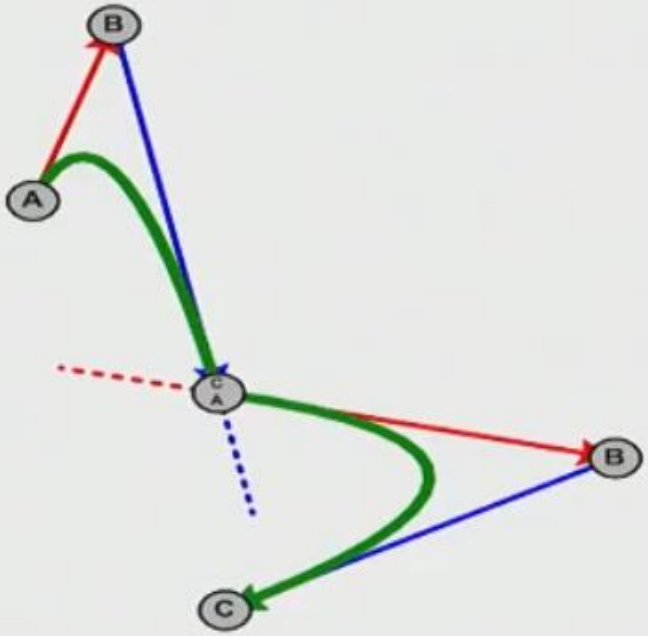
Topic 8

~47:00

Continuity



» Good continuity (C^1); connected **and** aligned



» Poor continuity (C^0); connected but not aligned

Catmull-Rom Splines

Topic 9

Special type of Cubic Hermit spline

Catmull-Rom Splines

Topic 9

~53:00

Catmull-Rom Splines

- A **Catmull-Rom spline** is just a cubic Hermite spline with special values chosen for the velocities at the start (**U**) and end (**V**) points of each section.
- You can also think of Catmull-Rom not as a type of spline, but as a technique for building cubic Hermite splines.
- Best application: curve-pathing through points

Checklist for today

Curves revisited

- Tweens recap
- Interpolation Lerp ~ Linear Bezier, Quadratic Bezier, Cubic Bezier
  - De Casteljau's algorithm
  - Non-uniformity in length per step, i.e., “velocity”
- Quadratic Bezier midpoint control
- Easing functions as Quadratic / Cubic Bezier
- Non-invasive curve splitting
- Cubic Hermite curves, explanation of control points
- Splines
  - “velocity” continuity
    - Quadratic vs. Cubic Bezier in spline editing
  - Catmull-Rom

Faculty of Mathematics and Physics
Charles University, Prague
7th November 2024

Gameplay Programming Level 05

Physics

Collisions and Mechanics

The backstage!

Largely based on:
Gregory, J. (2018). *Game engine architecture*. crc Press.

Physics in games

Short summary

INTRODUCTION

Physics in games
Short summary

Physics in games is about **mechanics** (most of the time)
https://en.wikipedia.org/wiki/Mechanics

"**Mechanics** (Greek: μηχανική) is the area of physics concerned with the motions of macroscopic objects. Forces applied to objects result in displacements, or changes of an object's position relative to its environment."

More specifically it deals with **rigid body dynamics**.
https://en.wikipedia.org/wiki/Rigid_body_dynamics

"… **rigid-body dynamics** studies the movement of systems of interconnected bodies under the action of external forces

Physics in games
Short summary

Things game physics system can do

- Collision detection, simulation of rigid bodies under forces
  + Triggers (triggering volumes)
- Springs, joints
- Ray and shape casting
- Cloth simulation
- Fluid simulation

Things you can use it for

- Movement of rigid bodies (bullets, boulders, object explosions, …, vehicles with realistic suspensions)
- Gameplay triggers
- Rag doll death animations / Powered version
- Dangling props
- Water surface simulations
- Audio propagation
- …

Peek into the History
of Physics in Games

Gaming Bolt, How Do Developers Implement Physics In Video Games?
https://www.youtube.com/watch?v=ZGqkbH_V_Zc

Physics in games
Short summary

Why **to have** physics in your games (examples)?

1. To reproduce real-life experience
   Flight Simulator, Gran Turismo, …
2. To allow for physics-based puzzles
   Bridge Builder, The Incredible Machine, Portal, …
3. To allow for emergent gameplay
   e.g. rocket jumps in shooters
4. To provide convincible virtual environment to play with (sandbox games)
   Minecraft (w/ Physics), Besiege, …

Physics in games
Short summary

Why to have physics in your games **under control**?

1. Not to deter players from the core experience
   e.g. story-driven games

2. Not to annoy players with misbehaving objects
   e.g. objects shooting into the sky because of penetrations

3. Not to mess with artistic composition
   e.g. physics unpredictability causing mess in the scene

Physics in games
Short summary

In a nutshell we want to:

- Move objects around by the means of forces / impulses
- Prevent object penetration
- Simulate
  - Collisions
  - Gravity
  - Joints
  - Springs
  - Buyoncy of objects
  - Fluids

=> We need 1) collision system, 2) physics (mechanics) system, we will label both as the physics system (PS) colloquially

Physics System

The Loop

High-level Loop of Physics Systems (PSs)

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. Detect collisions
5. Solve constraints
6. Notify the game loop (e.g., display results)

Physics System
The Loop

1. **Setup physics world (objects/properties/colliders)**

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. Detect collisions
5. Solve constraints
6. Notify the game loop (e.g., display results)

1.

In this step, we need to setup objects within the physical world specifying their properties and colliders. A game framework/engine will typically allow us to bind position and rotation in the game world with their physical counterparts so we do not need to rewrite position/rotation changes manually.

Physics System

The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. Detect collisions
5. Solve constraints
6. Notify the game loop (e.g., display results)

Once the PS is started, it typically works in fixed time steps to avoid the source of non-determinism. Maintaining the simulation not to go out of the hand is challenging already and you do not want to behave differently everytime you run it.

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. **Apply forces/impulses -> update velocities**
3. Update positions/rotations
4. Detect collisions
5. Solve constraints
6. Notify the game loop (e.g., display results)

2.

The physics step starts by applying all force / impulses either as the result within the simulation (e.g. gravity) or imposed from the outside (e.g. the gameplay code).

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. **Update positions/rotations**
4. Detect collisions
5. Solve constraints
6. Notify the game loop (e.g., display results)

3.
Then having velocities and angular velocities ready, we can make a small time step and move/rotate the objects in the physical world.

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. **Detect collisions**
5. Solve constraints
6. Notify the game loop (e.g., display results)

4.
After moving/rotating things, it's time to detect collisions between objects, typically in the form of penetrations, producing temporary constrains over the position/rotation changes.

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. Detect collisions
5. **Solve constraints**
6. Notify the game loop (e.g., display results)

5.
Those temporary constraints are then put together with other constraints as imposed e.g. by joints and they are all solved to receive the final position/rotation/velocities of objects.

Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. Detect collisions
5. Solve constraints
6. **Notify the game loop (e.g., display results)**

6.

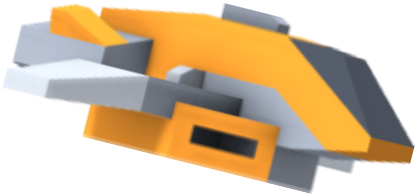
Last but not least, we need to notify the game loop that another "physics frame" has finished and new values have been crunched for the objects. This is a time to update "game world" counterparts of "physical objects".

Physics Systems

In a nutshell

"Physics" World and Game Objects

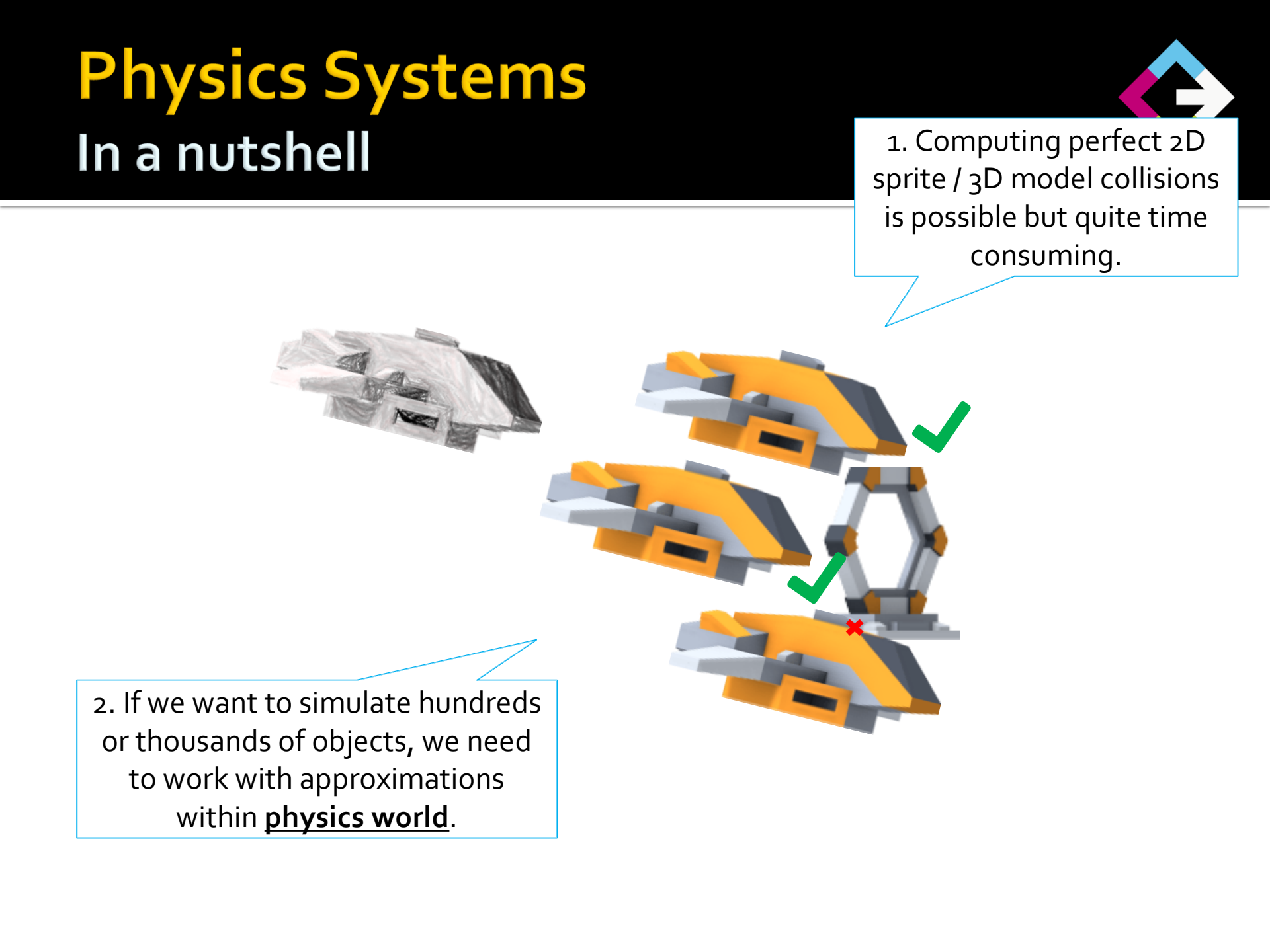
Physics Systems
In a nutshell



1. "Visualized" game world is full of fancy game objects.

2. We typically assume that all game objects have **rigid bodies** that cannot be deformed (cf. soft bodies, which are much more complex to simulate).

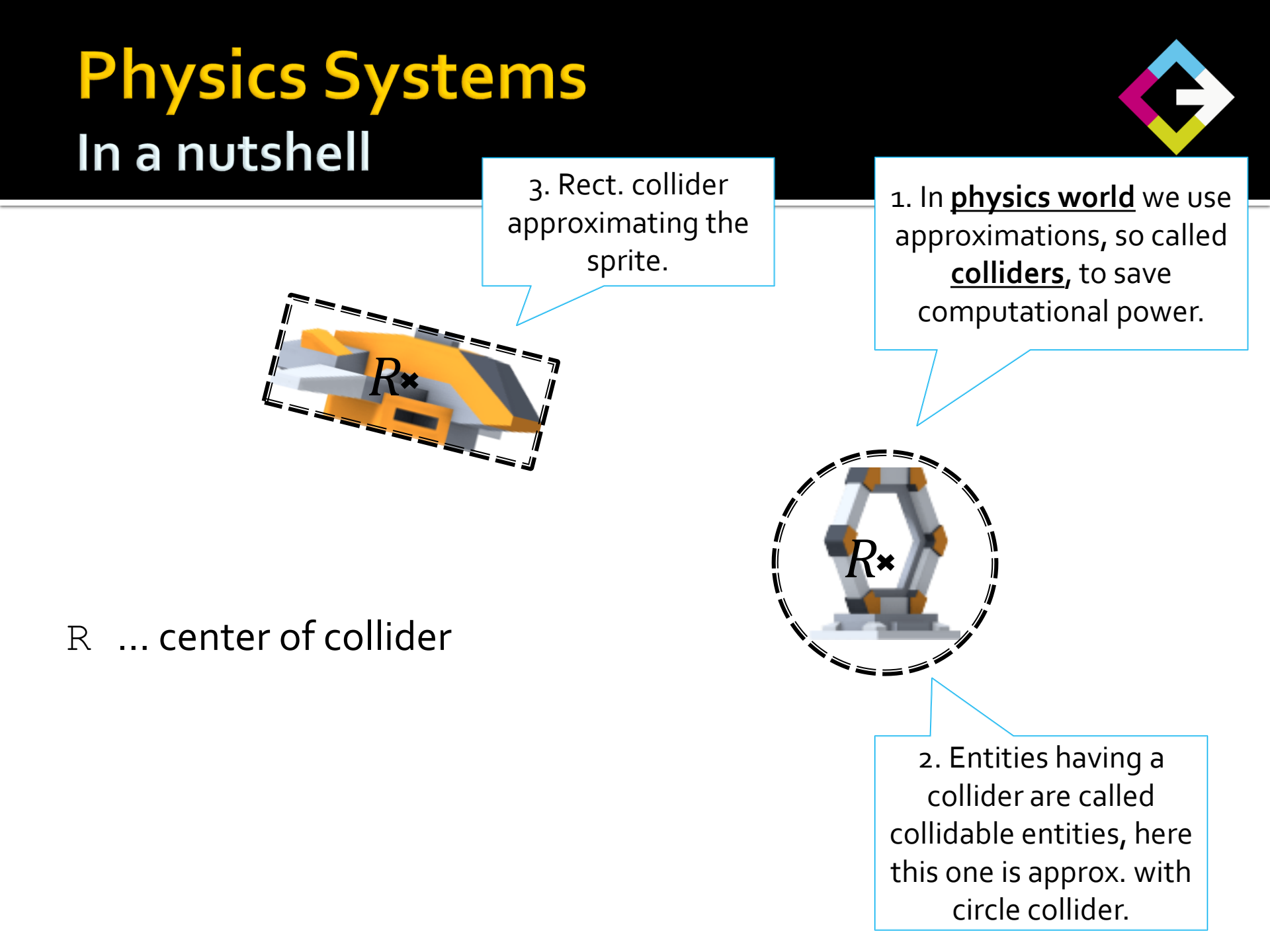
Physics Systems
In a nutshell



1. Computing perfect 2D sprite / 3D model collisions is possible but quite time consuming.

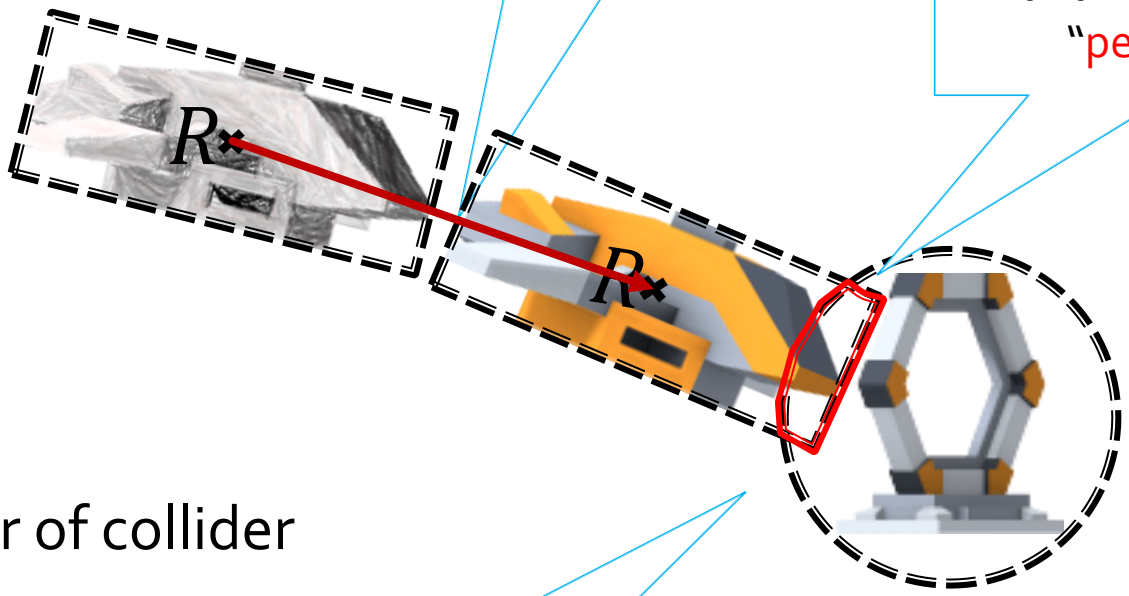
2. If we want to simulate hundreds or thousands of objects, we need to work with approximations within **physics world**.

Physics Systems
In a nutshell



`R` ... center of collider

Physics Systems
In a nutshell



1. Game objects can be moved by gameplay code if you wish to.

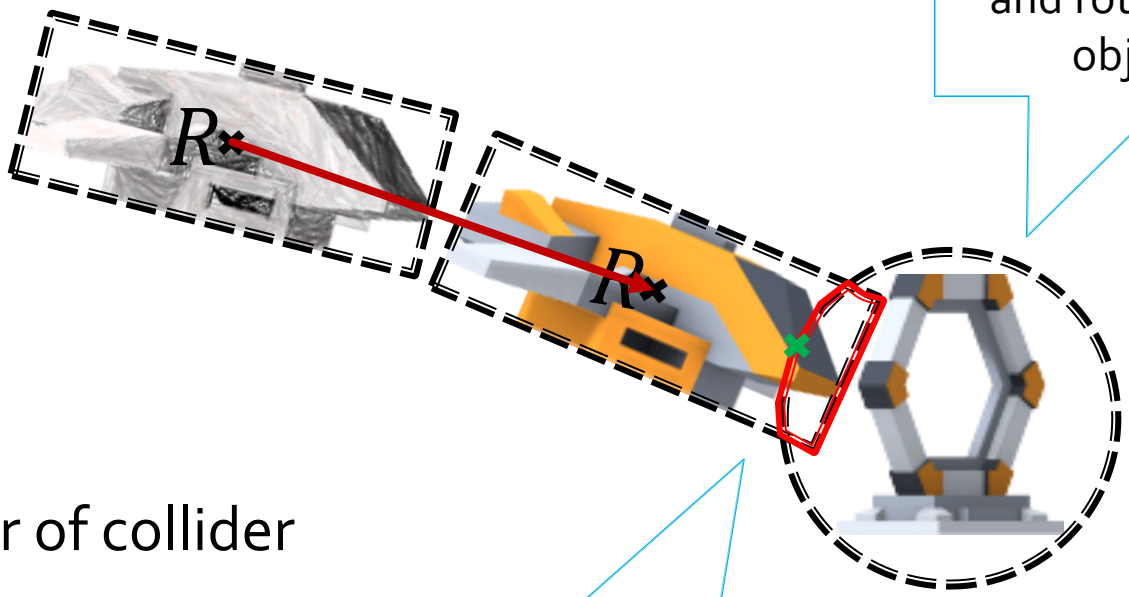
2. In which case, physics systems only care about collision computations and watches out for "penetrations".

`R` … center of collider

3. Once we detect the collision, the question is, how it should be resolved?

Physics Systems
In a nutshell

1. If we set this gate to be "immovable", then the PS will search for the first point of contact. Via backtracking the position and rotation of moving object linearly.

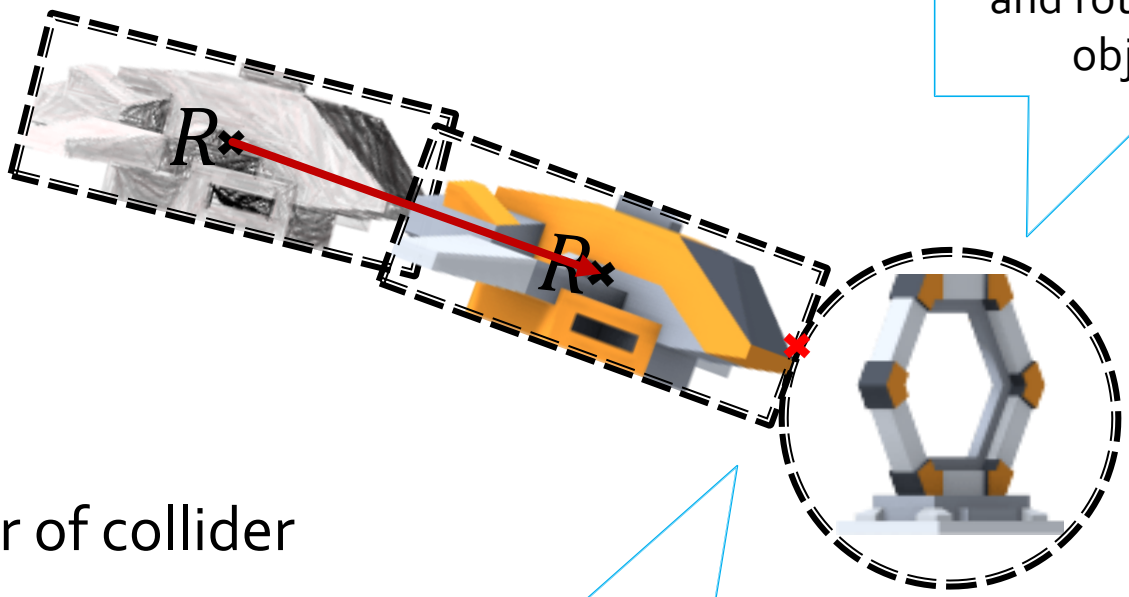


R … center of collider

2. And move the object back a bit perhaps rotate it a bit to conform to the constraints of collider shapes and thus preventing their penetration.

Physics Systems
In a nutshell

1. If we set this gate to be "immovable", then the PS will search for the first point of contact. Via backtracking the position and rotation of moving object linearly.



`R` ... center of collider

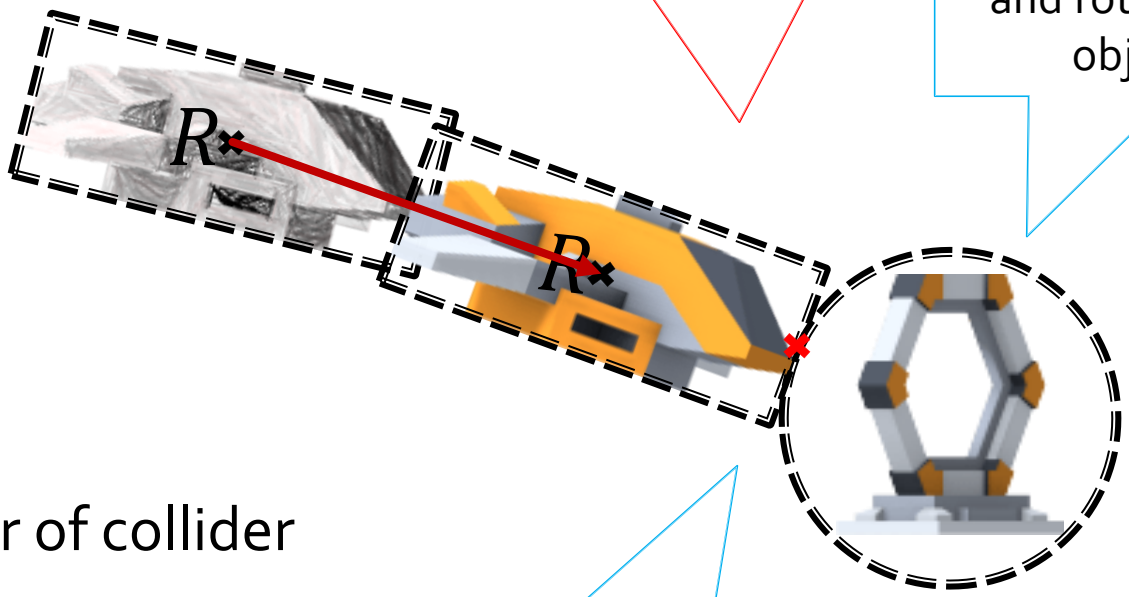
2. And move the object back a bit perhaps rotate it a bit to conform to the constraints of collider shapes and thus preventing their penetration.

3. But how does the PS know what position and rotation it should push the sprite back to?

Physics Syst
In a nutshell

4. Watch out for timing! You should not move your sprites multiple times in-between physics frames!

1. If we set this gate to be "immovable", then the PS will search for the first point of contact. Via backtracking the position and rotation of moving object linearly.



R … center of collider

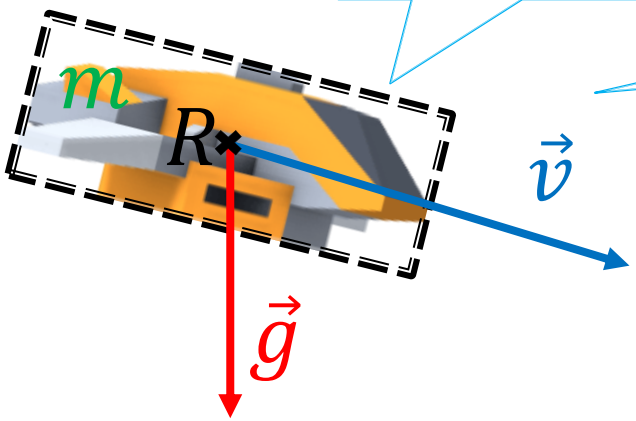
2. And move the object back a bit perhaps rotate it a bit to conform to the constraints of collider shapes and thus preventing their penetration.

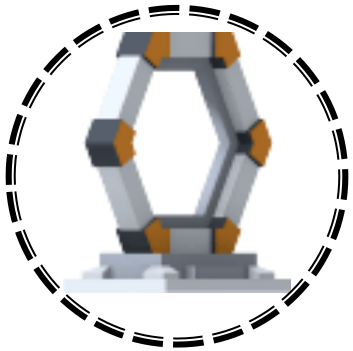
3. But how does the PS know what position and rotation it should push the sprite back to?

Physics Systems
In a nutshell

2. In which case, the PS needs more information about the object.

1. We might also want the objects to be moved by the PS itself.

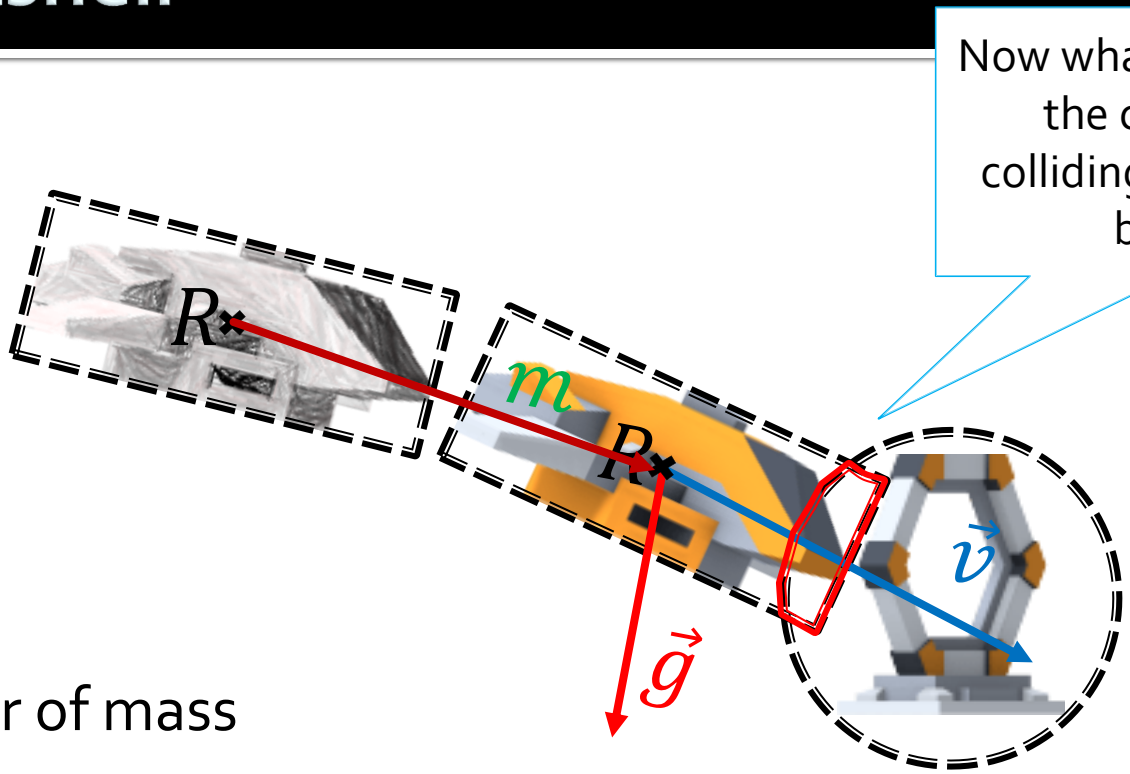




R … center of mass
m … mass
v … velocity
g … gravity

3. Each such a physical object is well-described using physical quantities and is subject to forces such as, e.g., a gravity. We assume the mass to be evenly distributed around centers of colliders. If you have more complex object you decompose it into multiple physical objects with fixed joints.

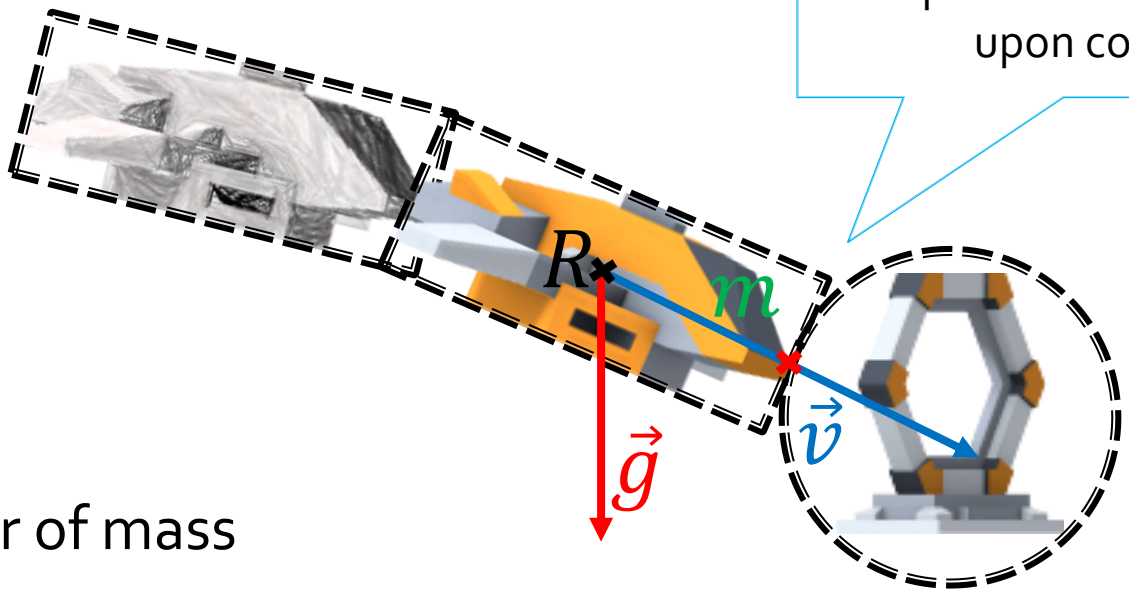
Physics Systems
In a nutshell



R … center of mass
m … mass
v … velocity
g … gravity

Physics Systems
In a nutshell

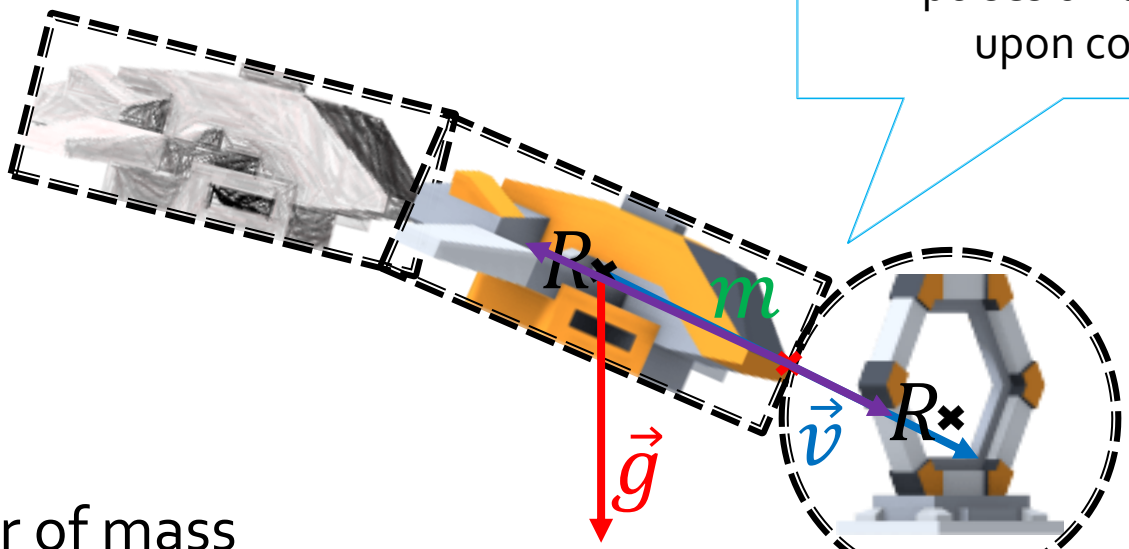
1. If both objects are setup as such, the PS uses the Law of conservation implied by 3rd Newton law to generate impulses on both objects upon collision.



R … center of mass
m … mass
v … velocity
g … gravity

Physics Systems
In a nutshell

1. If both objects are setup as such, the PS uses the Law of conservation implied by 3rd Newton law to generate impulses on both objects upon collision.



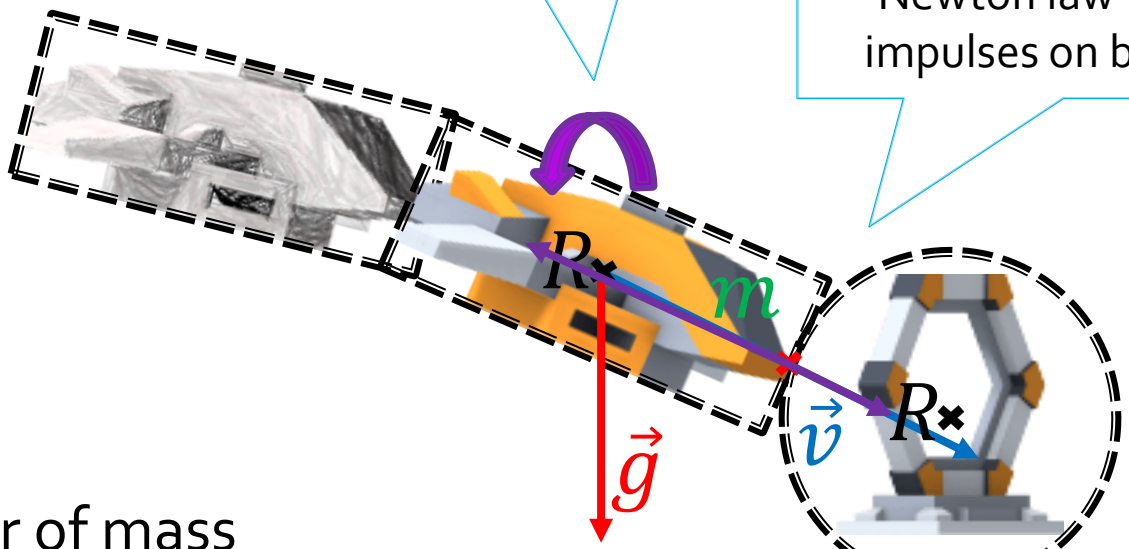
R … center of mass
m … mass
v … velocity
g … gravity

2. **Momentum** is a mass multiplied by velocity.
Impulse is the change of momentum.
Law of conservation state that momentum cannot be created or destroyed in a closed system. Thus we generate two **impulses** opposite to each other.

Physics S
In a nutshe

3. If the PS is allowed to change rotation of the object, it will also generate an **angular impulse**.

1. If both objects are setup as such, the PS uses Law of conservation implied by 3rd Newton law to generate impulses on both objects.

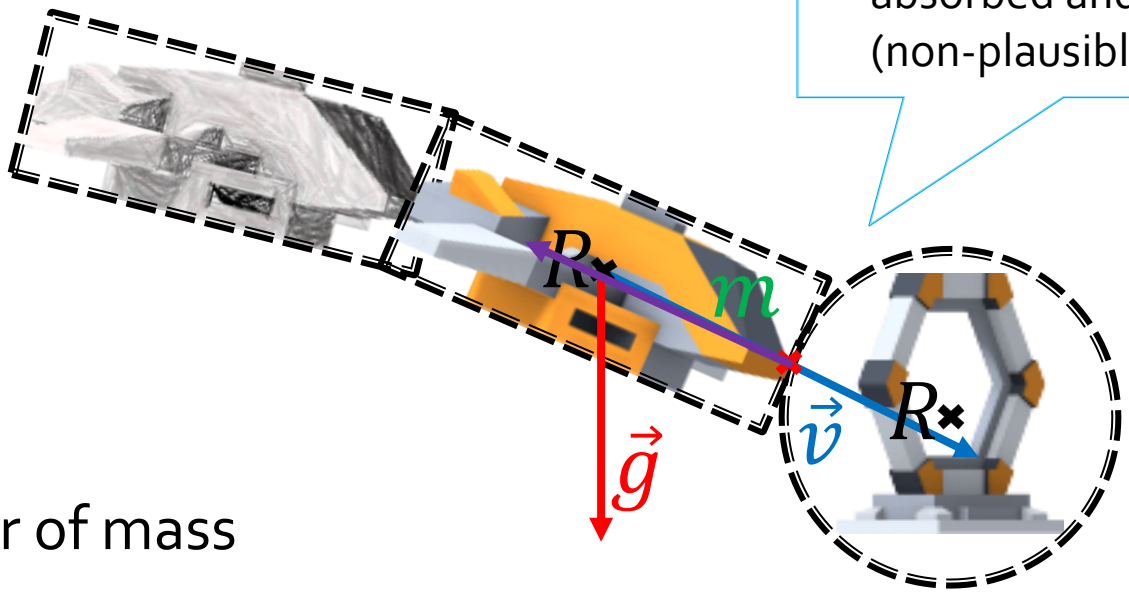


R … center of mass
m … mass
v … velocity
g … gravity

2. **Momentum** is a mass multiplied by velocity.
Impulse is the change of momentum.
Law of conservation state that momentum cannot be created or destroyed in a closed system. Thus we generate two **impulses** opposite to each other.

Physics Systems
In a nutshell

In case that one object is set to "immovable", the **impulse** generated for the star gate is absorbed and disappears (non-plausible behavior).



R ... center of mass
m ... mass
v ... velocity
g ... gravity

Physics Systems

In a greater detail

Diving deeper…

$\Rightarrow$ Rigid Body Dynamics

Newtonian Mechanics

Theory behind physics systems

Physics in games is about **classical mechanics** (most of the time)
https://en.wikipedia.org/wiki/Classical_mechanics

"**Classical**[note 1] **mechanics** is a physical theory describing the motion of macroscopic objects, from projectiles to parts of machinery, and astronomical objects, such as spacecraft, planets, stars and galaxies."

Classical mechanics (sometimes called Newtonian) has three subfields:

- Kinematics https://en.wikipedia.org/wiki/Kinematics
- (Analytical) Dynamics https://en.wikipedia.org/wiki/Analytical_dynamics
- Statics https://en.wikipedia.org/wiki/Statics

We are interested in dynamics most of the time.

Rigid Body Dynamics

In a greater detail

Most rigid body dynamics simulations operate in the MKS system of units:

- Distance is measured in meters, scalar [**m**]
- Mass is measured in kilograms, scalar [**k**g]
- Time is measured in seconds, scalar [**s**]

Hence the name MKS.

Other quantities:

- Linear velocity, vector [m/s]
- Acceleration, vector [m/s^2]
- Momentum, vector [kg.m/s]
- Force, vector [$kg.m/s^2$]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

In a greater detail

An **unconstrained** rigid body is one that:

- can translate freely along (all) two (in 2D) / three (in 3D) Cartesian axes
- can rotate freely about these one (in 2D) / three (in 3D) axes as well.

We say that such a body has three / six degrees of freedom (DOF).

The motion of such bodies can be separated into two **independent** components:

- **Linear dynamics** (motion ignoring all rotational effects)
- **Angular dynamics** (rotational motion)

=> We compute them separately and then combine

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

In a greater detail

Linear Dynamics

Rigid Body Dynamics
Linear dynamics

For the purposes of linear dynamics, an unconstrained rigid body acts as though all of its mass were concentrated at a single point known as the center of mass (abbreviated CM, or sometimes COM). We drop CM subscript in further text.

$$\mathbf{r}_{\mathrm{CM}} = \frac{\sum_{\forall i} m_i \mathbf{r}_i}{\sum_{\forall i} m_i} = \frac{\sum_{\forall i} m_i \mathbf{r}_i}{m}$$

r – position in space, a position vector (x,y) in 2D, (x,y,z) in 3D
$\mathbf{r}_{cm}$ – position of center of mass
m – mass, scalar [kg]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics
Linear velocity

Linear velocity of a rigid body defines the speed and direction in which the body's CM is moving [m/s].

$$\mathbf{v}(t) = \frac{d\mathbf{r}(t)}{dt} = \dot{\mathbf{r}}(t)$$

The **dot** here marks the first derivative with respect to time.

Differentiating a vector is the same as differentiating each component independently.

$$v_x(t) = \frac{dr_x(t)}{dt} = \dot{r}_x(t)$$

Derivative is relative change of a quantity in a given amount of time here.

r – position in space, vector
?$_x$ – x component of **?**
v – velocity, vector [m/s]
t – time, scalar [s]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Linear acceleration

Linear acceleration is the first derivative of linear velocity with respect to time, or the second derivative of the position of a body's CM versus time [m/s²].

$$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt} = \dot{\mathbf{v}}(t) = \frac{d^2\mathbf{r}(t)}{dt^2} = \ddot{\mathbf{r}}(t)$$

r – position in space, vector
$\mathbf{r}_x$ – x component of **r**
v – velocity, vector [m/s]
t – time, scalar [s]
a – acceleration, vector [m/s²]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Force

A force is defined as anything that causes an object with mass to accelerate or decelerate, a vector. The final force affecting the object is computed as a (vector) sum of respective forces.

$$\mathbf{F}_{\text{net}} = \sum_{i=1}^{N} \mathbf{F}_i$$

$\mathbf{F}_{net}$ – final force affecting the object, vector [N = kg.m/s^2]
$\mathbf{F}_i$ – i-th force affecting the object

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Newton's laws

1st Newton's law

An object either remains at rest or continues to move at a constant velocity, unless acted upon by a force.

2nd Newton's law

The sum of the forces F on an object is equal to the mass m of that object multiplied by the acceleration a of the object: $F = m.a$

3rd Newton's law

When one body exerts a force on a second body, the second body simultaneously exerts a force equal in magnitude and opposite in direction on the first body.

F – final force affecting the object [N = kg.m/s²]
m – mass, scalar [kg]
a – acceleration, vector [m/s²]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

2nd Newton's law

2nd Newton law

The sum of the forces F on an object is equal to the mass m of that object multiplied by the acceleration a of the object: `F = m.a`

This assumes the mass to be constant.

$$\mathbf{F}(t) = m\mathbf{a}(t) = m\ddot{\mathbf{r}}(t)$$

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

2nd Newton's law

2nd Newton law

If **mass is not constant**, as would be the case for a rocket whose fuel is being gradually used up and converted into energy, equation needs to be corrected.

$$\mathbf{F}(t) = \frac{d\mathbf{p}(t)}{dt} = \frac{d(m(t)\mathbf{v}(t))}{dt}$$

Where `p(t)` is linear momentum.

$$\mathbf{p}(t) = m\mathbf{v}(t)$$

F – final force affecting the object, vector [N = kg.m/s²]
m – mass, scalar [kg]
p – linear momentum, vector [kg.m/s]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Solving the Equations of Motion

A force can be constant, or it can be a function of time as shown above. A force can also be a function of the position of the body, its velocity, or any number of other quantities. So, in general, the expression for force should really be written as follows:

$$\mathbf{F}(t, \mathbf{r}(t), \mathbf{v}(t), \ldots) = m\mathbf{a}(t)$$

$$\mathbf{F}(t, \mathbf{r}(t), \dot{\mathbf{r}}(t), \ldots) = m\ddot{\mathbf{r}}(t)$$

r – position in space
F – final force affecting the object, vector [N = kg.m/s²]
m – mass, scalar [kg]
v – velocity, vector [m/s]
t – time, scalar [s]
a – acceleration, vector [m/s²]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Solving the Equations of Motion

We can see the position equation as a function of time, an ordinary differential equation (ODE).

$$\ddot{\mathbf{r}}(t) = \frac{1}{m}\mathbf{F}(t, \mathbf{r}(t), \dot{\mathbf{r}}(t))$$

We "can" use explicit Euler method (linear interp.) to solve this:

$$\mathbf{r}(t_2) = \mathbf{r}(t_1) + \mathbf{v}(t_1)\Delta t$$

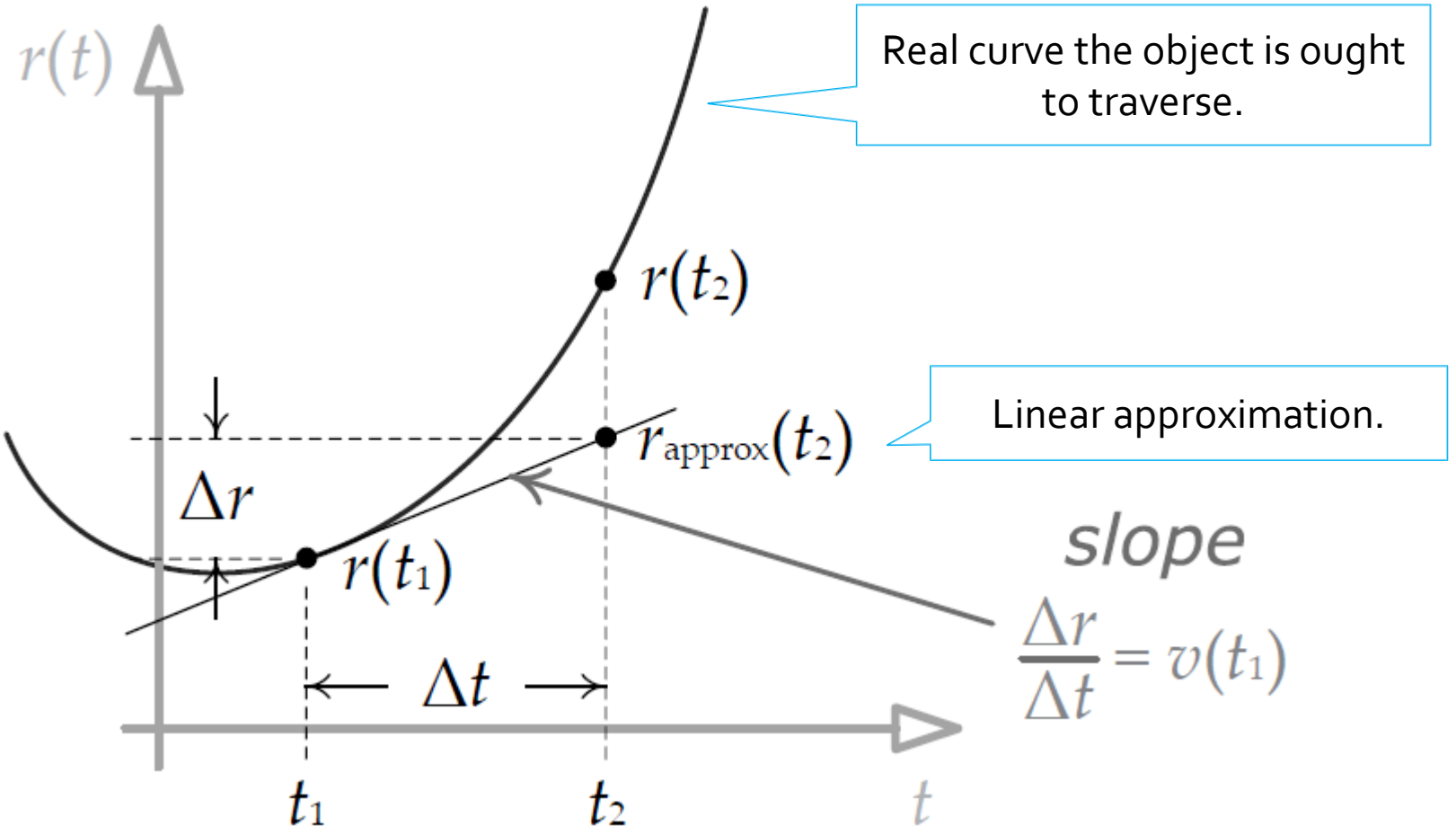
$$\mathbf{a}(t) = \frac{\mathbf{F}_{\text{net}}(t)}{m} = \dot{\mathbf{v}}(t)$$

$$\mathbf{v}(t_2) = \mathbf{v}(t_1) + \frac{\mathbf{F}_{\text{net}}(t)}{m}\Delta t$$

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Limitation of Explicit Euler



Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics
Sought Solution

A numerical solution to an ordinary differential equation has three important and interrelated properties:

Convergence. As the time step Δt tends toward zero, does the approximate solution get closer and closer to the real solution?

Order. Given a particular numerical approximation to the solution of an ODE, how "bad" is the error? (Typically proportional to Δt)

Stability. Does the numerical solution tend to "settle down" over time?

Nowadays, Verlet integration (its velocity variant) is typically used, but there are many more approaches to this.

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics
Velocity Verlet

If we know `a(t1)` ...

$$\mathbf{a}(t_1) = \frac{1}{m}\mathbf{F}\big(t_1, \mathbf{r}(t_1), \mathbf{v}(t_1)\big)$$

We do the following:

1. Calculate $\mathbf{r}(t_1 + \Delta t) = \mathbf{r}(t_1) + \mathbf{v}(t_1)\Delta t + \frac{1}{2}\mathbf{a}(t_1)\Delta t^2$.
2. Calculate $\mathbf{v}(t_1 + \frac{1}{2}\Delta t) = \mathbf{v}(t_1) + \frac{1}{2}\mathbf{a}(t_1)\Delta t$.
3. Determine $\mathbf{a}(t_1 + \Delta t) = \mathbf{a}(t_2) = \frac{1}{m}\mathbf{F}\big(t_2, \mathbf{r}(t_2), \mathbf{v}(t_2)\big)$.
4. Calculate $\mathbf{v}(t_1 + \Delta t) = \mathbf{v}(t_1 + \frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{a}(t_1 + \Delta t)\Delta t$.

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

In a greater detail

Angular Dynamics in 2D

Rigid Body Dynamics

Angular dynamics in 2D

In 2D, angular dynamics works almost identically to linear dynamics. For each linear quantity, there's an angular analog, and the mathematics works out quite neatly.

Every rigid body can be treated as a thin sheet of material. All linear motion occurs in the xy-plane, and all rotations occur about the z-axis.

The orientation of a rigid body in 2D is fully described by an angle θ, measured in radians relative to some agreed-upon zero rotation.

θ – rotation, scalar [rad]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics
Angular speed and acceleration

Angular speed measures the rate at which a body's rotation angle changes over time: ω [rad/s]
Angular acceleration, denoted α(t) and measured in radians per second squared [rad/s²].

Angular:	Linear:
$\omega(t) = \frac{d\theta(t)}{dt} = \dot{\theta}(t)$	$\mathbf{v}(t) = \frac{d\mathbf{r}(t)}{dt} = \dot{\mathbf{r}}(t)$
$\alpha(t) = \frac{d\omega(t)}{dt} = \dot{\omega}(t) = \ddot{\theta}(t)$	$\mathbf{a}(t) = \frac{d\mathbf{v}(t)}{dt} = \dot{\mathbf{v}}(t) = \ddot{\mathbf{r}}(t)$

θ – rotation, scalar [rad]
ω – angular speed, scalar [rad/s]
α – angular acceleration [rad/s²]

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics
Moment of intertia

The rotational equivalent of mass is a quantity known as the **moment of inertia**.

Just as mass describes how easy or difficult it is to change the linear velocity of a point mass, the moment of inertia measures how easy or difficult it is to change the angular speed of a rigid body about a particular axis.

Since we're focusing on two-dimensional angular dynamics right now, the axis of rotation is always **z**, and a body's moment of inertia is a simple scalar value denoted by **I**.

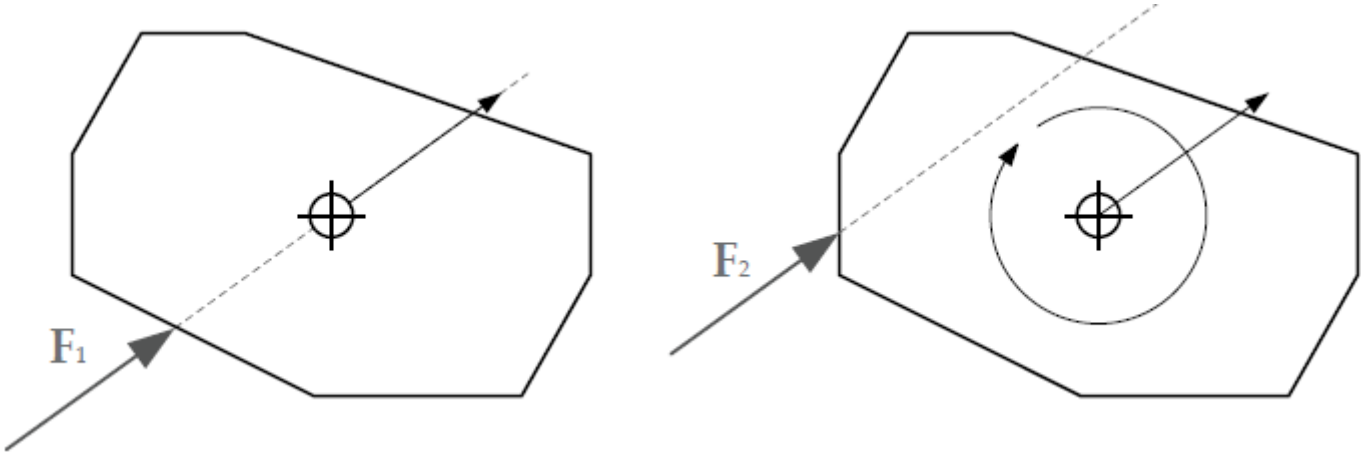
θ – rotation, scalar [rad]
ω – angular speed, scalar [rad/s]
α – angular acceleration [rad/s²]
I – moment of intertia

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Torque

In general, forces can be applied at arbitrary points on a body. If the line of action of a force passes through the body's center of mass, then the force will produce linear motion only, as we've already seen. Otherwise, the force will introduce a rotational force known as a *torque* in addition to the linear motion it normally causes, denoted **N**.



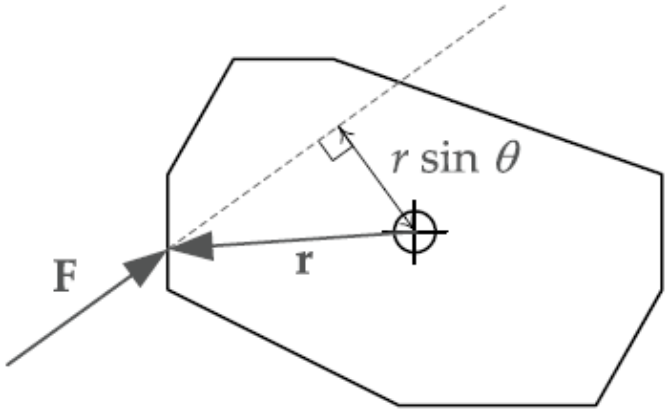
Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Torque

Torque **N** is calculated by taking the cross product between a force's point of application in body space **r** (i.e., relative to the center of mass) and the force vector **F**. The vectors are shown here in two dimensions for ease of illustration; if it could be drawn, the torque vector would be directed into the slide.

$$\mathbf{N} = \mathbf{r} \times \mathbf{F}$$



Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

Torque

Torque is related to angular acceleration and moment of inertia in much the same way that force is related to linear acceleration and mass.

Angular:	Linear:
$N_z(t) = I\alpha(t) = I\dot{\omega}(t) = I\ddot{\theta}(t)$	$\mathbf{F}(t) = m\mathbf{a}(t) = m\dot{\mathbf{v}}(t) = m\ddot{\mathbf{r}}(t)$
$N_{\text{net}}(t) = I\dot{\omega}(t)$	$\mathbf{F}_{\text{net}}(t) = m\dot{\mathbf{v}}(t)$
$\omega(t) = \dot{\theta}(t)$	$\mathbf{v}(t) = \dot{\mathbf{r}}(t),$

Approximate explicit Euler solutions are (in reality we use, e.g., Verlet instead):

$\omega(t_2) = \omega(t_1) + I^{-1}N_{\text{net}}(t_1)\Delta t$	$\mathbf{v}(t_2) = \mathbf{v}(t_1) + m^{-1}\mathbf{F}_{\text{net}}(t_1)\Delta t$
$\theta(t_2) = \theta(t_1) + \omega(t_1)\Delta t$	$\mathbf{r}(t_2) = \mathbf{r}(t_1) + \mathbf{v}(t_1)\Delta t.$

Gregory, J. (2018). Game engine architecture. crc Press.

Rigid Body Dynamics

In a greater detail

Angular Dynamics in 3D
is beyond the scope here…

Physics Systems

In a greater detail

Going deeper… next time…

$\Rightarrow$ Collision Detection

Collision Detection

In a greater detail (follow-up lecture)

In the follow-up lecture, we will study the collision detection using the following superb article:
https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects

Topics for the exam:

- Collision detection – broad / narrow phase
- AABB, Sort and sweep, Dynamic bounding volume trees
- Convex vs. Concave shapes
- Separating axes theorem
- Gilbert-Johnson-Keerthi Algorithm
- Expanding Polytope Algorithm
- Tunneling and Continuous Collision Detection

Physics Systems

In a greater detail

Constraints (e.g. joints)

Constraints
… and joints

An unconstrained rigid body has six degrees of freedom (DOF) in 3D: It can translate in three dimensions, and it can rotate about the three Cartesian axes. ***Constraints*** restrict an object's motion, reducing its degrees of freedom either partially or completely.

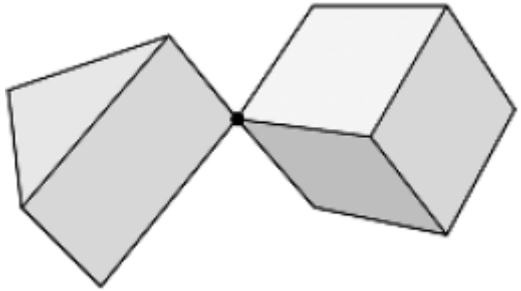
- Swinging chandelier (point-to-point constraint);
- A door that can be kicked (hinge constraint);
- A vehicle's wheel assembly (axle constraint with damped springs for suspension);
- A train or a car pulling a trailer (stiff spring/rod constraint);
- A rope or chain (chain of stiff springs or rods)

Gregory, J. (2018). Game engine architecture. crc Press.

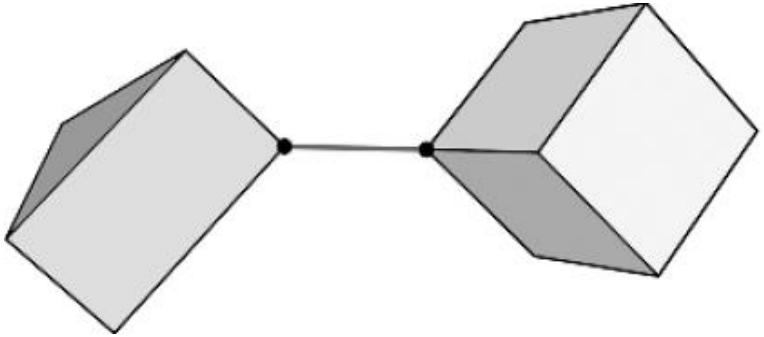
Constraints

Point-to-point, Stiff spring

Point-to-point constraint requires that a point on body A aligns with a point on body B.



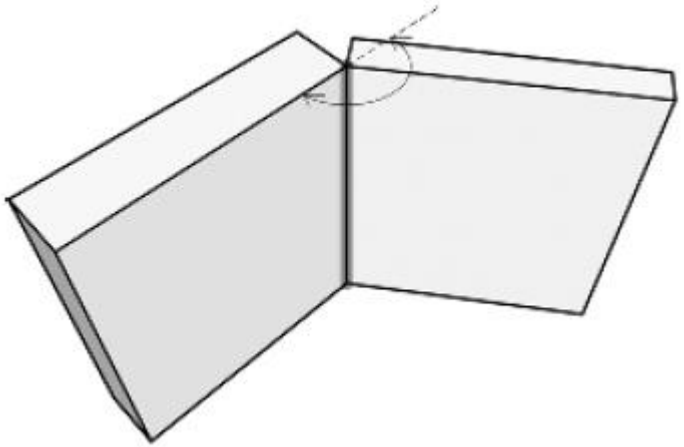
A stiff spring constraint requires that a point on body A be separated from a point on body B by a user-specified distance.



Gregory, J. (2018). Game engine architecture. crc Press.

Constraints
Hinge

A hinge constraint limits rotational motion to only a single degree of freedom, about the hinge's axis. An unlimited hinge acts like an axle, allowing the constrained object to complete an unlimited number of full rotations. It's common to define limited hinges that can only move through a predefined range of angles about the one allowed axis.



Gregory, J. (2018). Game engine architecture. crc Press.

Constraints
Chains

Long chains of linked bodies are sometimes difficult to simulate in a stable manner because of the iterative nature of the constraint solver. A constraint chain is a specialized group of constraints with information that tells the constraint solver how the objects are connected. This allows the solver to deal with the chain in a more stable manner than would otherwise be possible.

Gregory, J. (2018). Game engine architecture. crc Press.

Constraints

The Solver

A constraint solver is essentially an iterative algorithm that attempts to satisfy a large number of constraints simultaneously by minimizing the error between the actual positions and rotations of the bodies in the physics world and their ideal positions and rotations as defined by the constraints. As such, constraint solvers are essentially iterative error-minimization algorithms.

Out of scope of the course; more info can be found at: https://www.toptal.com/game/video-game-physics-part-iii-constrained-rigid-body-simulation

Gregory, J. (2018). Game engine architecture. crc Press.

Physics Systems

Phew!

Physics in games

Where to next?

Video Game Physics Introduction
https://www.toptal.com/game/video-game-physics-part-i-an-introduction-to-rigid-body-dynamics
https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects
https://www.toptal.com/game/video-game-physics-part-iii-constrained-rigid-body-simulation

Game Physics Lecture at Utrecht University:
http://www.cs.uu.nl/docs/vakken/mgp/2018-2019/index.html

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Material has been produced within and supported by the project „Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“ kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.

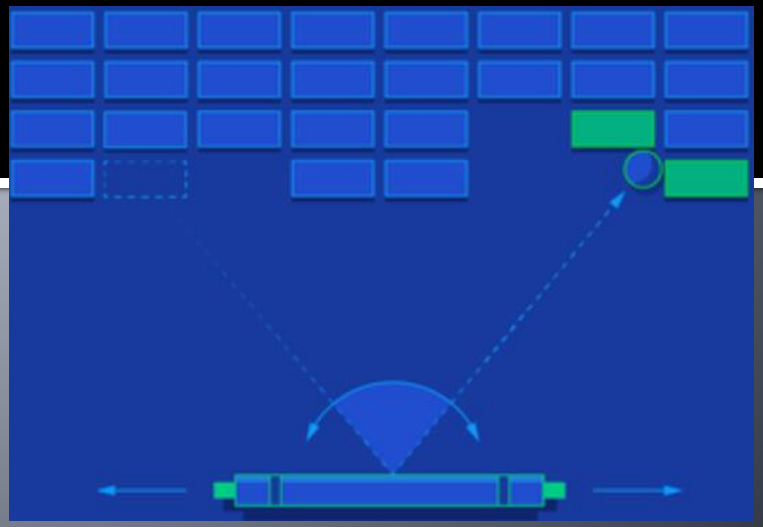
Faculty of Mathematics and Physics
Charles University, Prague
14th November 2024

Gameplay Programming Level 06

Physics

Collision Detection

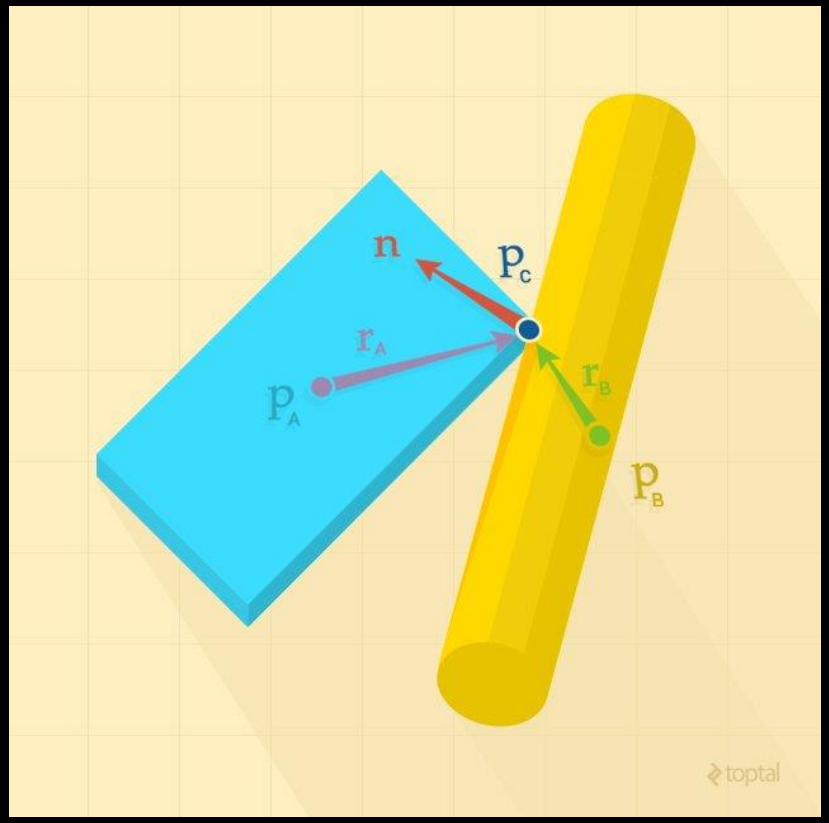
The backstage!



(Almost) totally based on:
https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects

COLLISION DETECTION

What?



Physics System
The Loop

1. Setup physics world (objects/properties/colliders)

`Loop until` game/level finished with a time-step *t*:

2. Apply forces/impulses -> update velocities
3. Update positions/rotations
4. **Detect collisions // <-- COVERED TODAY**
5. Solve constraints
6. Notify the game loop (e.g., display results)

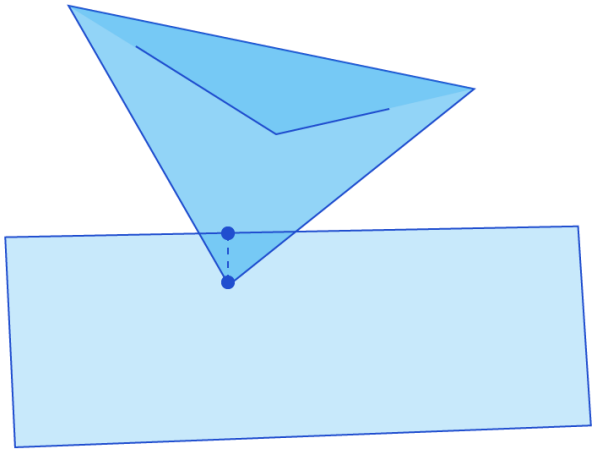
http://realtimecollisiondetection.net/

Collision Detection

The Problem

Given n geometric shapes (planar in 2D, volumetric in 3D), return pairs, which are *colliding* together with their contact points.

We say that two objects *colliding* or "are in collision" if there is at least on point that belongs to each of them.



Two objects, which are "in collision" and their contact points.

Collision Detection

Complexity

Number of collisions is $O(n^2)$, where n is number of objects. Fine geometry-to-geometry check requires $O(m^2)$ operations, where m is number of faces (3D) or segments (2D) of two objects in question.

Collision Detection
Algorithm skeleton

Naïve-1 algorithm computed every frame:

```
1.   for (x,y) in combinations(N,N):
2.     cp = contact_points(x,y)
3.     if cp is not empty:
4.       result += (x, y, cp)
5.   return result
```

Detailed geometry-to-geometry comparisons. So called the "**narrow phase**" of collision detection

Collision Detection
Algorithm skeleton

Naïve-2 algorithm computed every frame:

```
1.   for (x,y) in combinations(N,N):
2.      if might_be_colliding(x,y):
3.        cp = contact_points(x,y)
4.        if cp is not empty:
5.          result += (x, y, cp)
6.   return result
```

Simpler collision check that rules out some of non-colliding pairs.

Collision Detection
Algorithm skeleton

Broad/Narrow-phase algorithm computed every frame:

```
1.   for (x,y) in might_collide(N,N):
2.      cp = contact_points(x,y)
3.      if cp is not empty:
4.        result += (x, y, cp)
5.   return result
```

So called the "**broad phase**" of collision detection.

Collision Detection
Algorithm skeleton

Broad/Mid/Narrow-phase algorithm computed every frame:

```
1.  for (x,y) in might_collide(N,N):
2.    X,Y = decompose(x),decompose(y)
3.    for (x',y') in might_collide(X x Y):
4.      cp = contact_points(x',y')
5.      if cp is not empty:
6.        result += (x, y, cp)
7.  return result
```

So called the "**mid-phase**" of collision detection. Decomposes objects into sub-objects that might collide.

Collision Detection

Algorithm skeleton

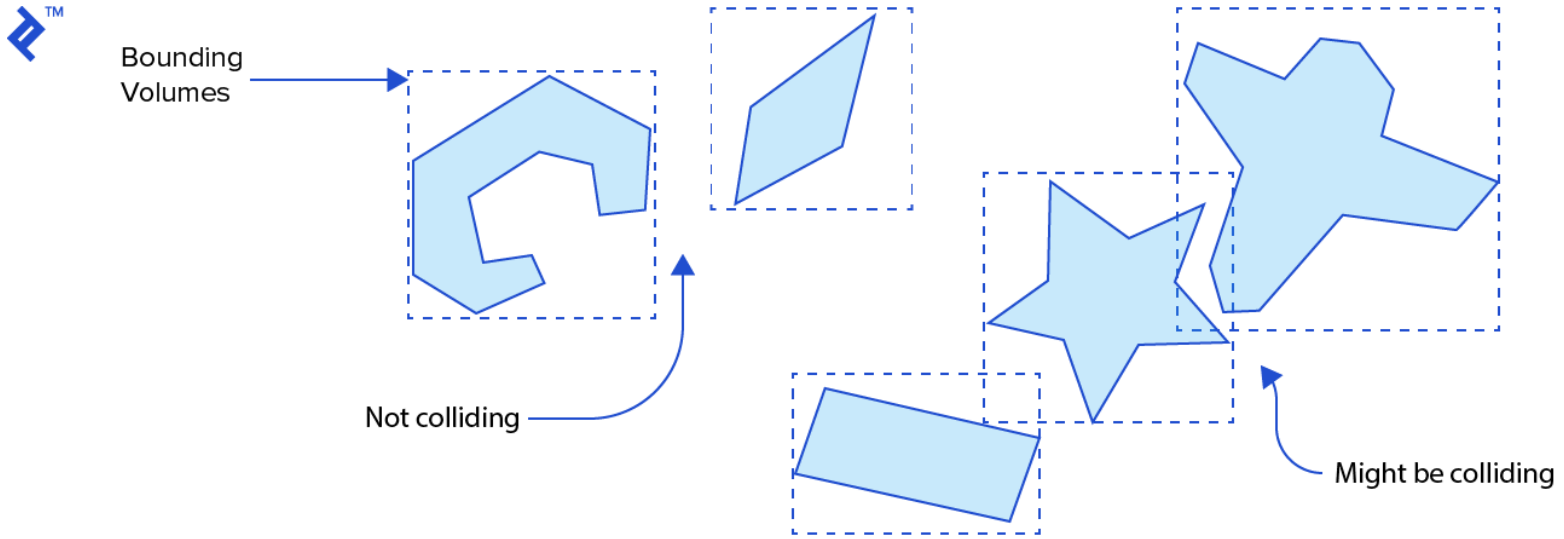
Somewhat-used-in-reality algorithm computed every frame:

```
1.  X = changed_objects(N)
2.  mc = update_might_collide(X, mc)
3.  for grp in mc:
4.    for (x,y) in combinations(grp, grp):
5.      X,Y = decompose(x),decompose(y)
6.      for (x',y') in might_collide(X x Y):
7.        cp = contact_points(x',y')
8.        if cp is not empty:
9.          result += (x, y, cp)
10. return result
```

Objects "do not move much" between frames; update created data structures from previous frame.

Collision Detection

Phases



Number of collisions is $O(n^2)$ => perform complex checks only for objects that **might** collide.

Collision detection split to two (three) phases:

1. Broad – rule out pairs that cannot collide at all
2. Mid – decompose objects into (simpler) sub-objects
3. Narrow – perform expensive geometry checks

COLLISION DETECTION

Broad phase

Collision Detection

Broad phase

For broad phase, we can use any technique that…

… rules out a collision for any two objects fast

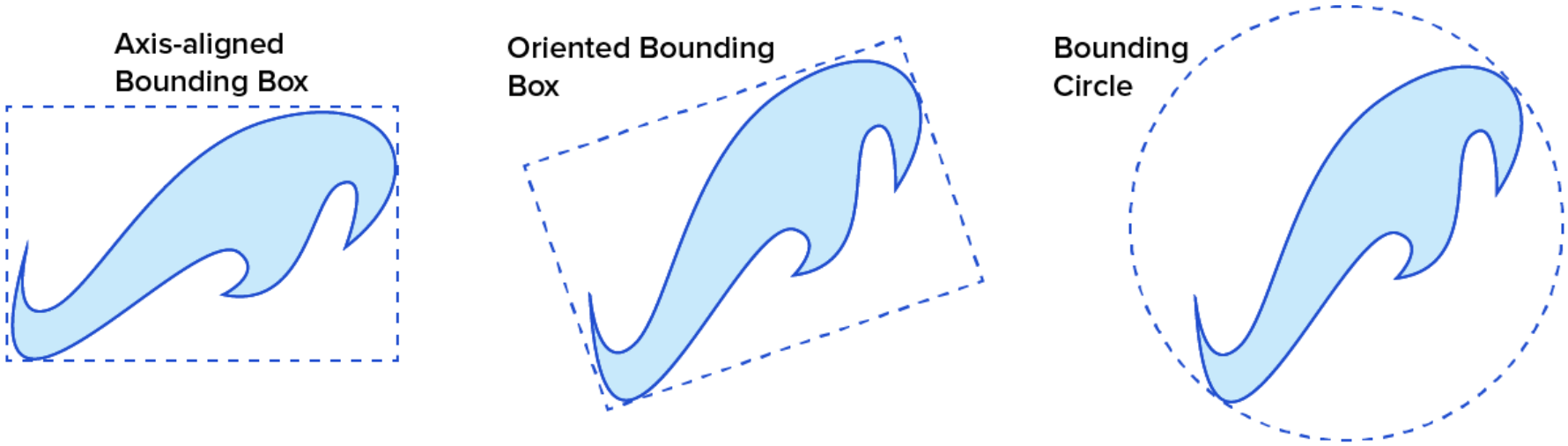
- less than $O(n)$, where ***n*** is total number of objects, ideally $O(1)$
- I.e., also not relying on the complexity of their geometry

Popular choices:

- Bounding boxes
- Sort and Sweep
- (Hierarchical) Uniform grids / Bounding volume trees

Collision Detection

Broad phase: bounding boxes

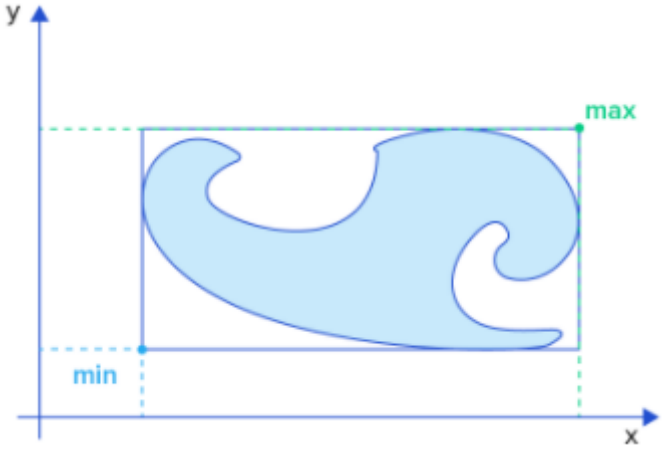


Encompass objects with more simple shapes and check whether they intersect.

Popular types: AABB, OBB, BC

Collision Detection – Broad Phase

AABB tests



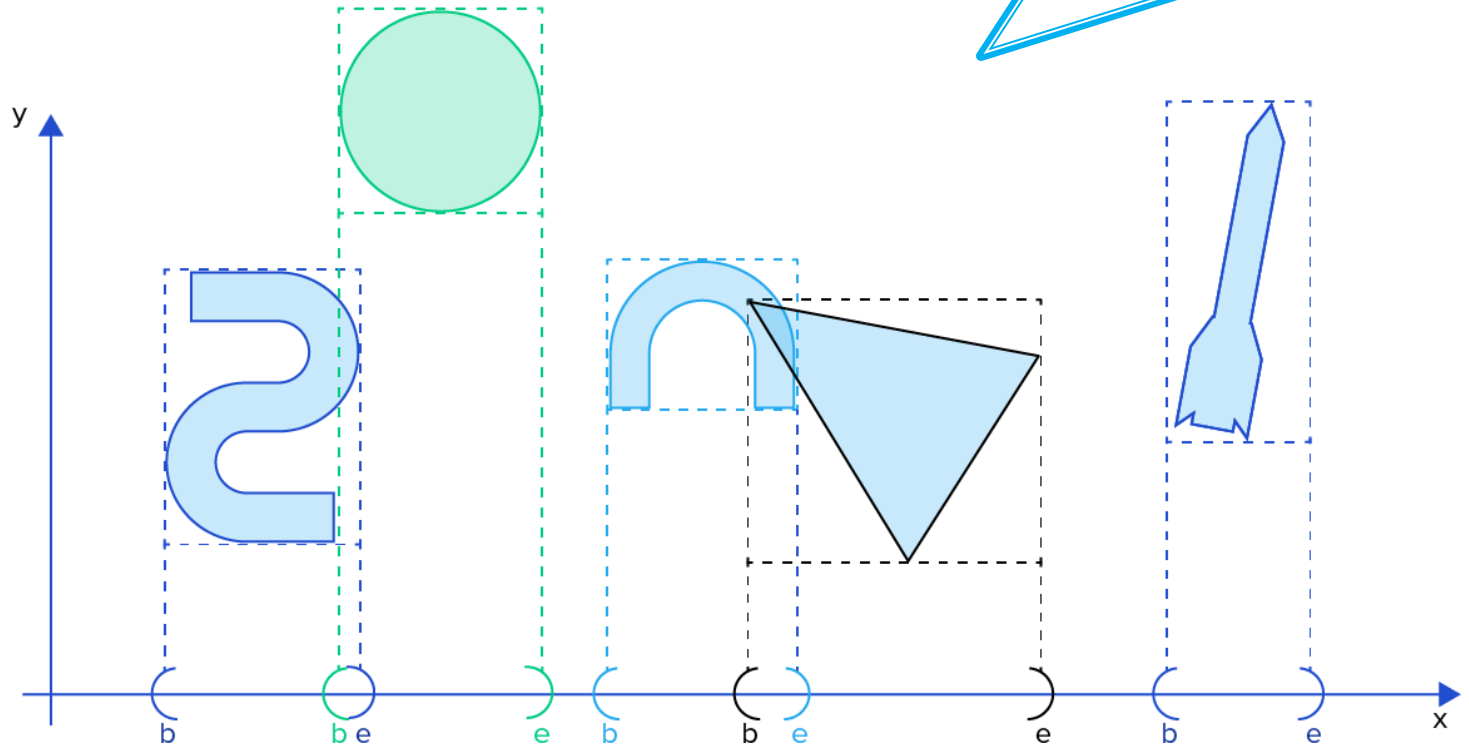

```
BOOL TestAABBOverlap(AABB* a, AABB* b) {
    float d1x = b->min.x - a->max.x;
    float d1y = b->min.y - a->max.y;
    float d2x = a->min.x - b->max.x;
    float d2y = a->min.y - b->max.y;
    if (d1x > 0.0f || d1y > 0.0f) return FALSE;
    if (d2x > 0.0f || d2y > 0.0f) return FALSE;
    return TRUE;
}
```

More interectsion, Point / Segment vs. AABB and AABB sweeping available at:
https://noonat.github.io/intersect/

Collision Detection – I

Sort and Sweep

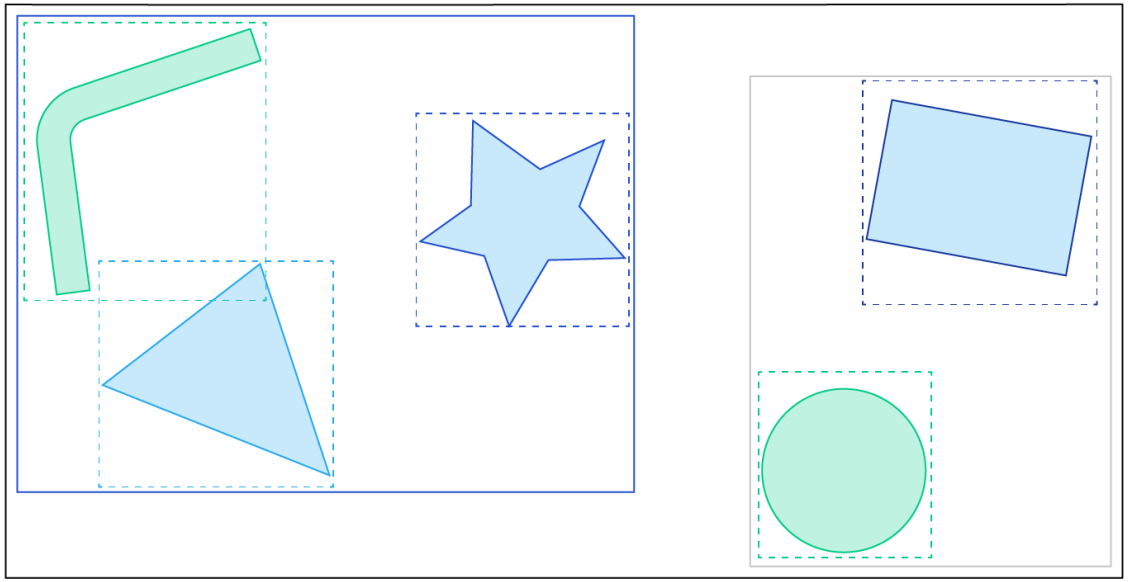
Store between frames, when object moves/rotates use insert sort to change its position on the axis. This update is close to O(n).

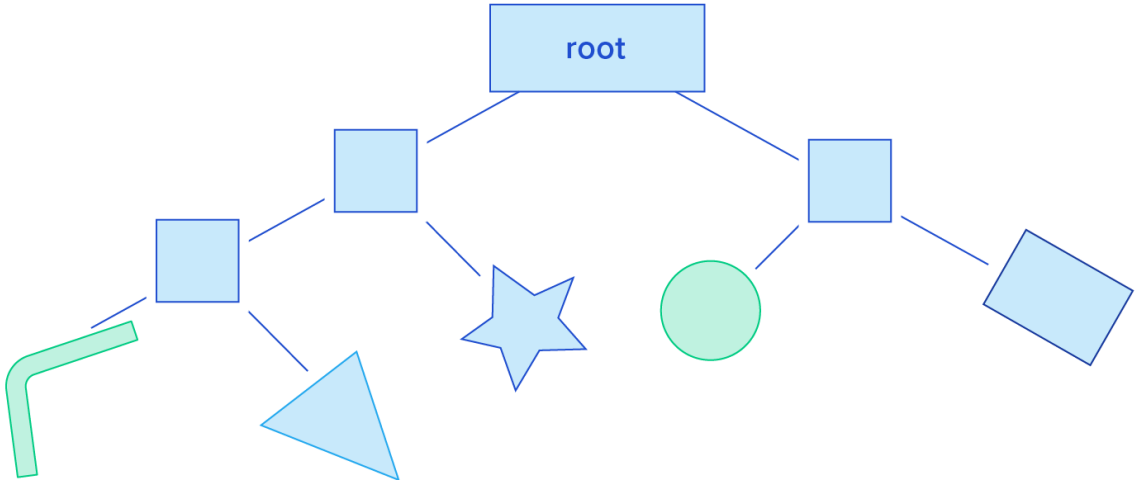


1. Project AABB onto axis ~ O(n)
2. Sort points ~ O(n.log n)
3. Iterate through the list, any “bb” encountered means possible intersection ~ O(n)

Collision Detection – Broad Phase

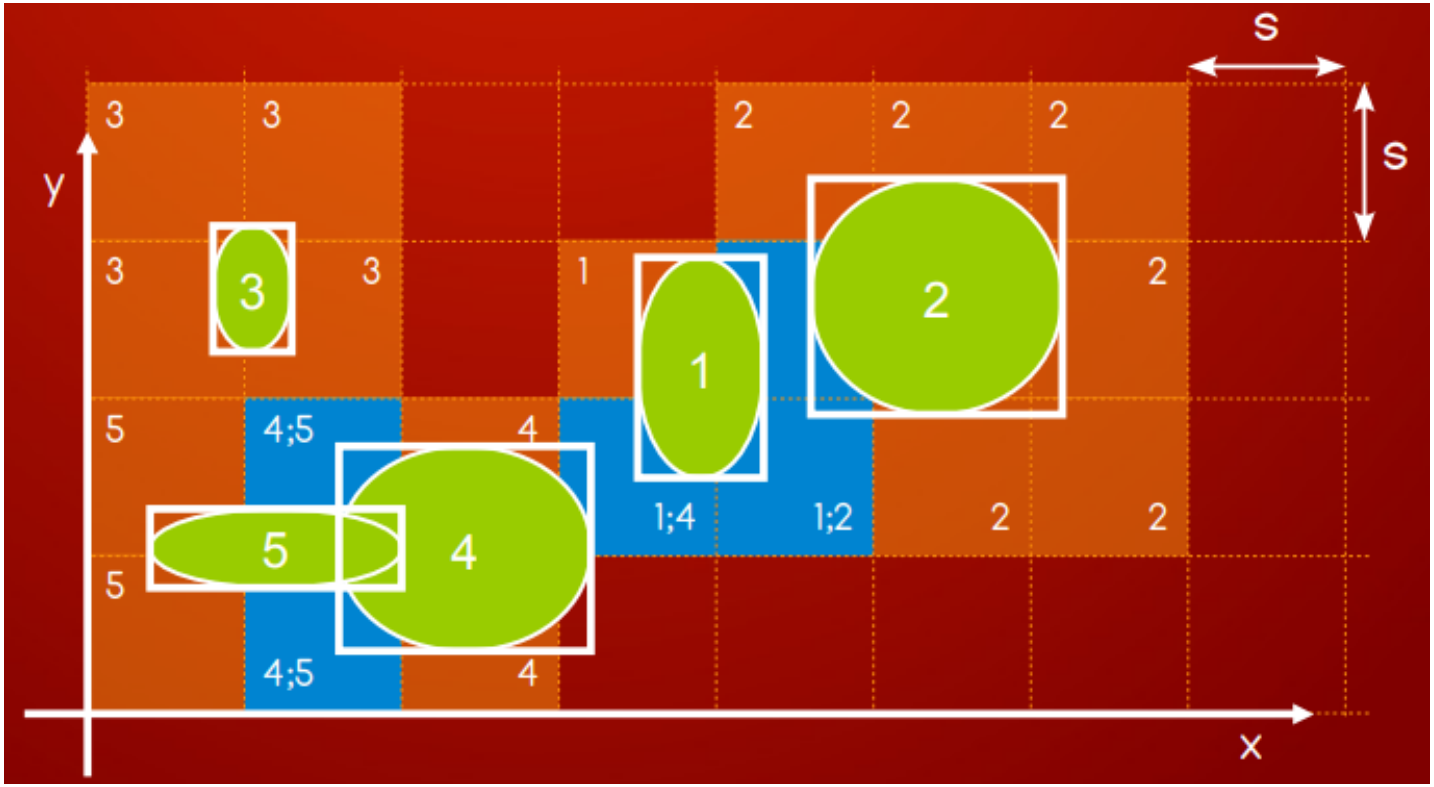
Dynamic Bounding Volume Trees (DBVT)





Collision Detection – Broad Phase

Uniform Grids



Cell color coding:

- Red – no objects
- Orange – single object
- Blue – multiple objects

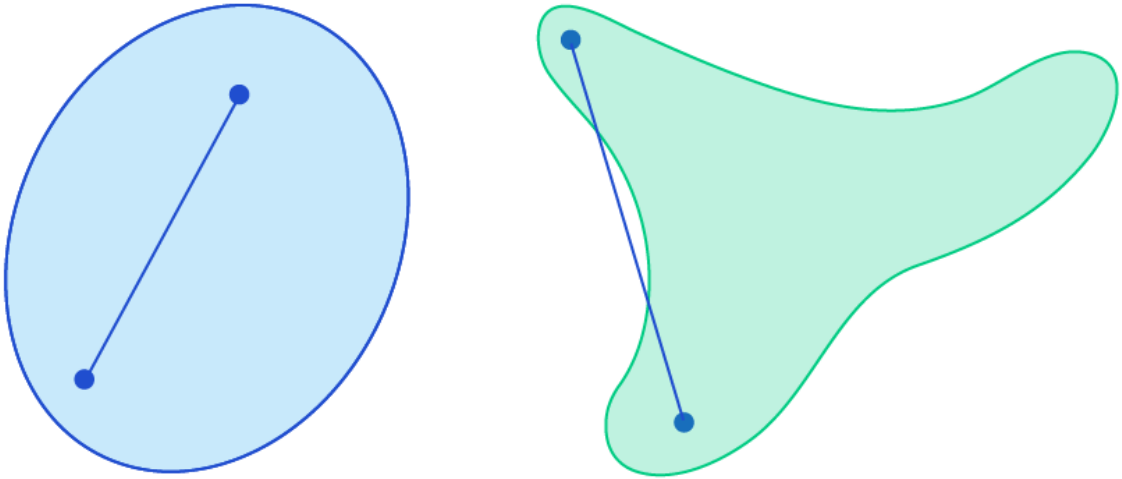
https://dai.fmph.uniba.sk/upload/8/87/Ca15_lesson05.pdf

COLLISION DETECTION

Mid phase

Collision Detection – Narrow Ph.

Convex vs. concave shapes



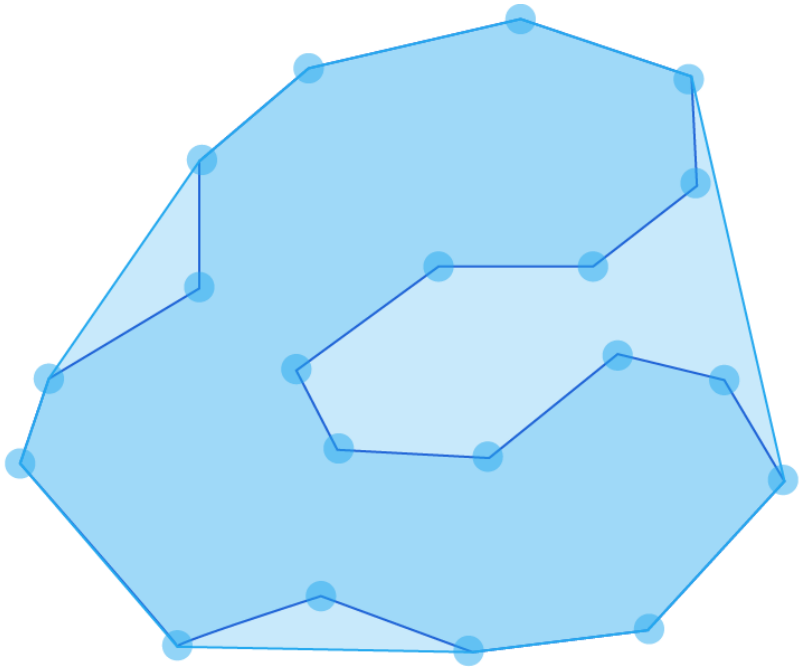
Convex

Concave

Many intersection algos for convex shapes, not many for concave. Concave can be either either covered into convex hull or decomposed to convex parts.

Collision Detection – Narrow Ph.

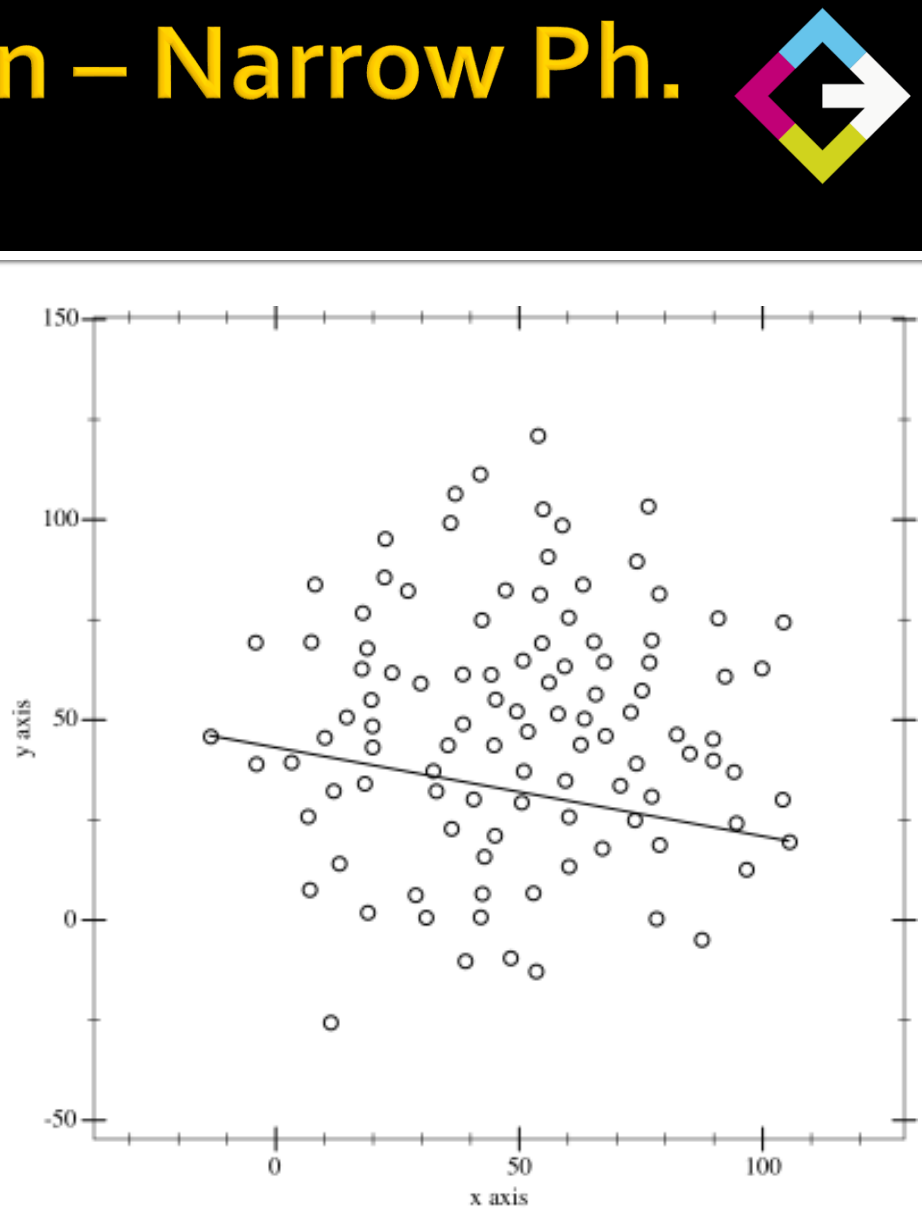
Convex hull



Concave shape covered into convex hull.
Convex hull can be found by quickhull algorithm in $O(n.\log n)$.
Ghost collision may appear, but are typically acceptable.

Collision Detection – Narrow Ph.

Quickhull algorithm



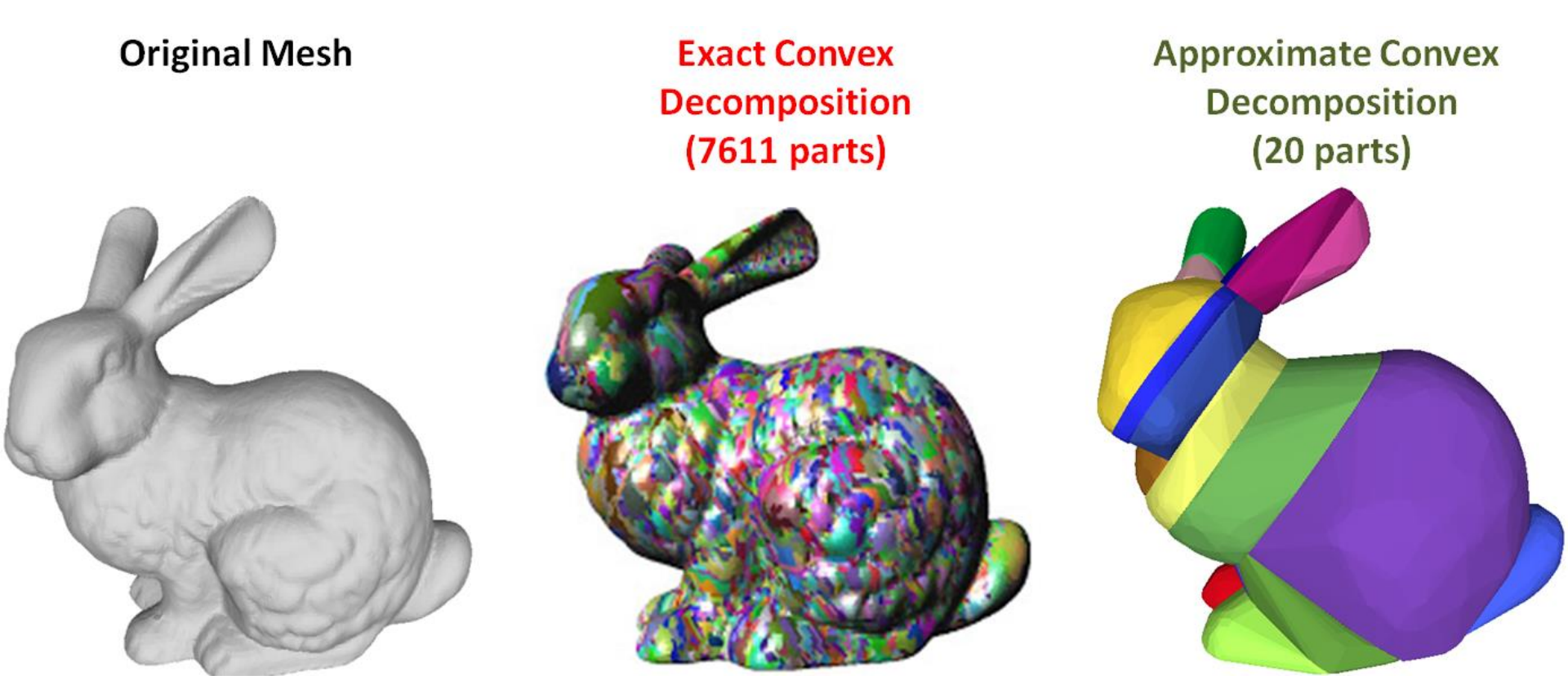
The **QuickHull algorithm** is a **Divide and Conquer algorithm** similar to QuickSort.

```
QH(points A)
1. X = min-x{A}, Y = max-x{A}
2. result ← XY
3. Line XY split points from A
   into two groups G1 and G2
3. QH-Recurse(G1, P1-P2)
4. QH-Recurse(G2, P1-P2)

QH-Recurse(points G, line XY)
3. Z = furthest a ∈ G from XY
4. result ← split XY into XZ and YZ
5. G = G \ { points inside XYZ }
6. New lines XZ and YZ
7. New subgroups G_XZ and G_YZ; each
   group contains points outside XZ
   and YZ respectively
8. QH-Recurse(G_XZ, XZ)
9. QH-Recurse(G_YZ, YZ)
```

Image source: https://iq.opengenus.org/quick-hull-convex-hull/

Collision Detection – Narrow Ph.

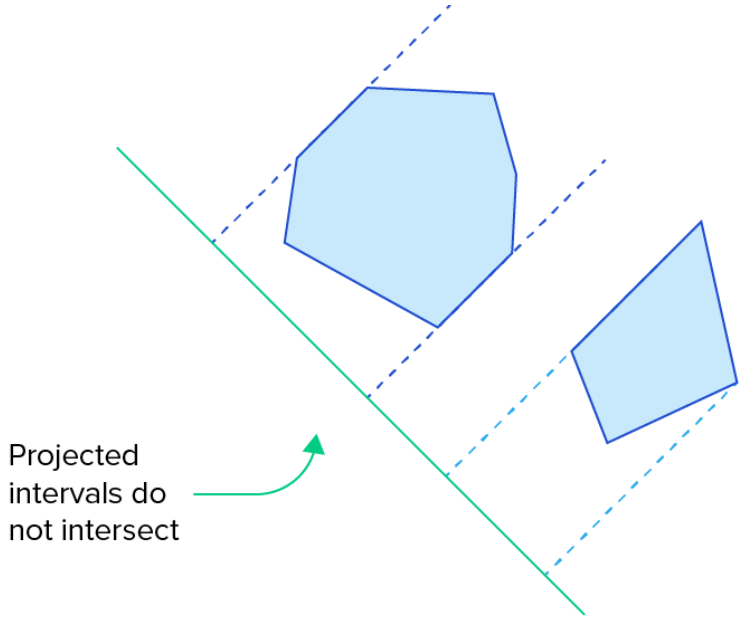
Convex decomposition

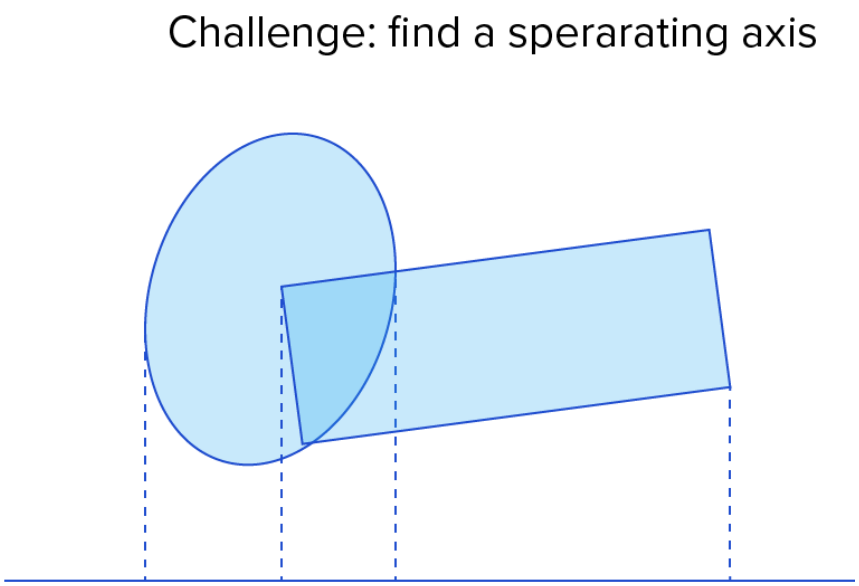


Concave shape (polyhedra) can be decomposed into convex one, e.g., by V-HACD.

Collision Detection – Mid Ph.

The separating axes theorem

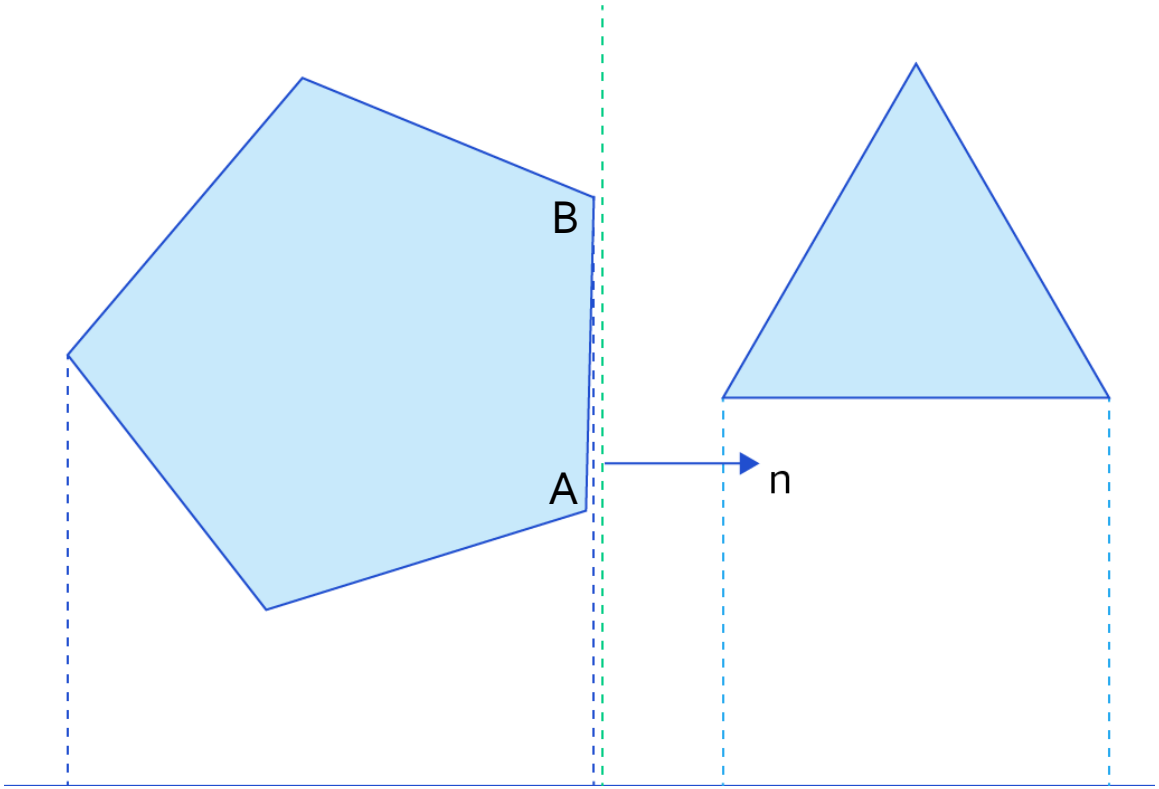




The separating axis theorem (SAT) states that two convex shapes are not intersecting if and only if there exists at least one axis where the orthogonal projections of the shapes on this axis do not intersect.

Collision Detection – Mid Ph.

The separating axes theorem



To find a separating axis, we can use edge/face normals.

2D: for edge A-B and vertex of the other shape V we test:

$(V - A).N > 0 \implies$ V is in front of an edge

COLLISION DETECTION
Narrow phase
Gilbert-Johnson-Keerthi Algorithm

Psst, a great video here:

A Strange But Elegant Approach to a Surprisingly Hard Problem (GJK Algorithm)

https://www.youtube.com/watch?v=ajv46BSqcK4

Main part is between 2:02-26:20

Collision Detection – Narrow Ph.

Gilbert-Johnson-Keerthi Algorithm

Concepts to understand for the exam:

- Minkowski sum and difference
- Origin in Minkowski difference?
- Simplex
- Support functions
- Broad steps of GJK algorithm
- Triple product
- How to determine a point passed an origin?
  - Voronoi regions

COLLISION DETECTION
Expanding Polytope Algorithm

A good video here:

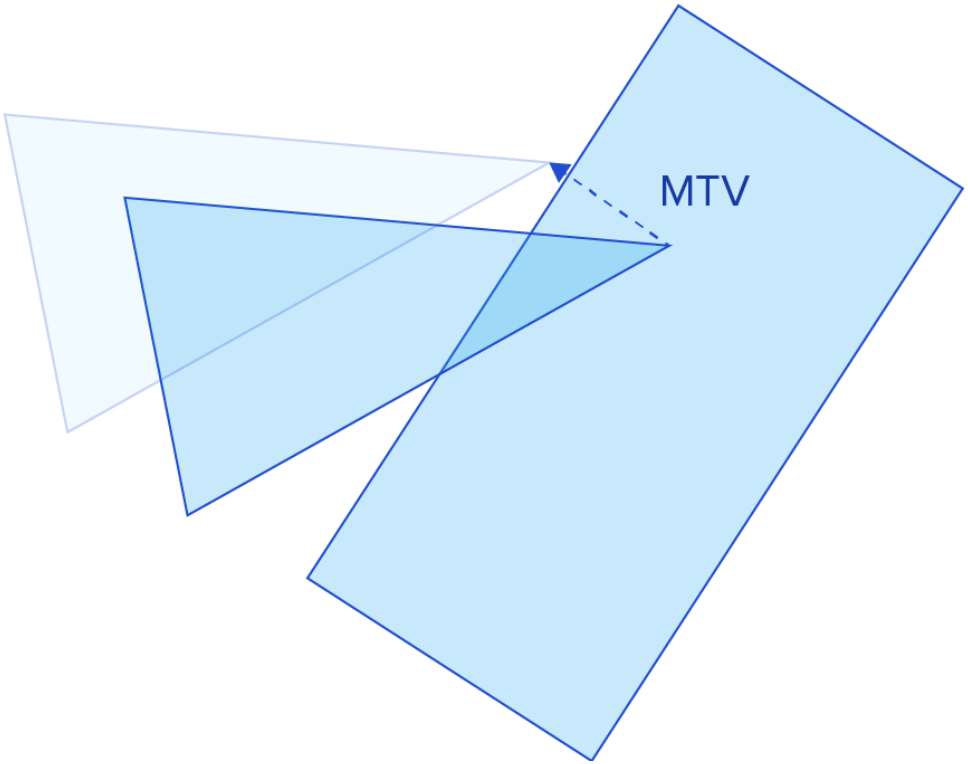
EPA Explanation & Implementation

https://www.youtube.com/watch?v=0XQ2FSz3EK8

Main part between 0:22 and 2:54

Collision Detection – Narrow Ph.

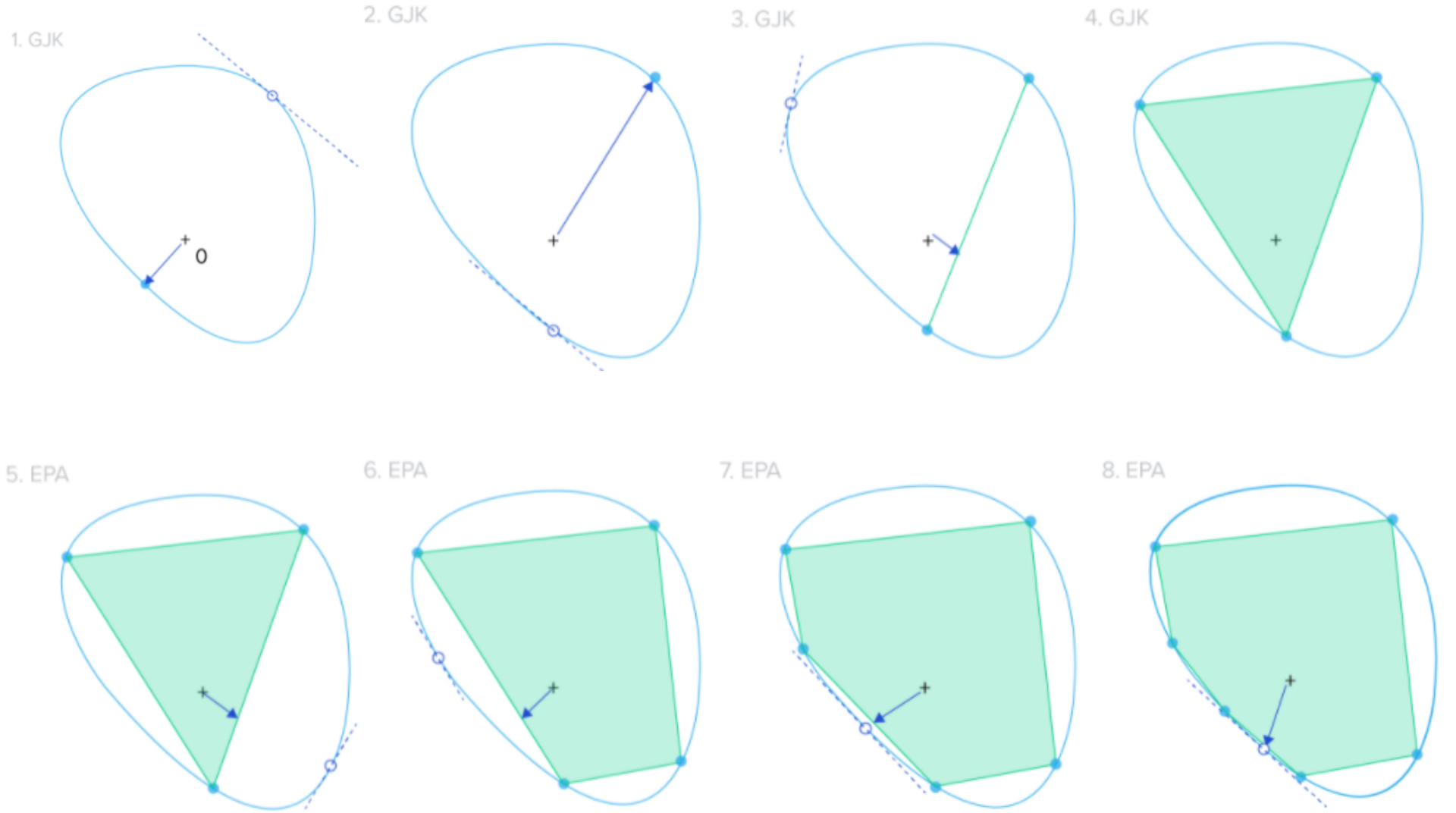
Expanding Polytope Algorithm



The **penetration depth** is the length of the **minimum translation vector** (MTV), which is the smallest vector along which we can translate an intersecting shape to separate it from the other shape.

Collision Detection – Narrow Ph.

Expanding Polytope Algorithm

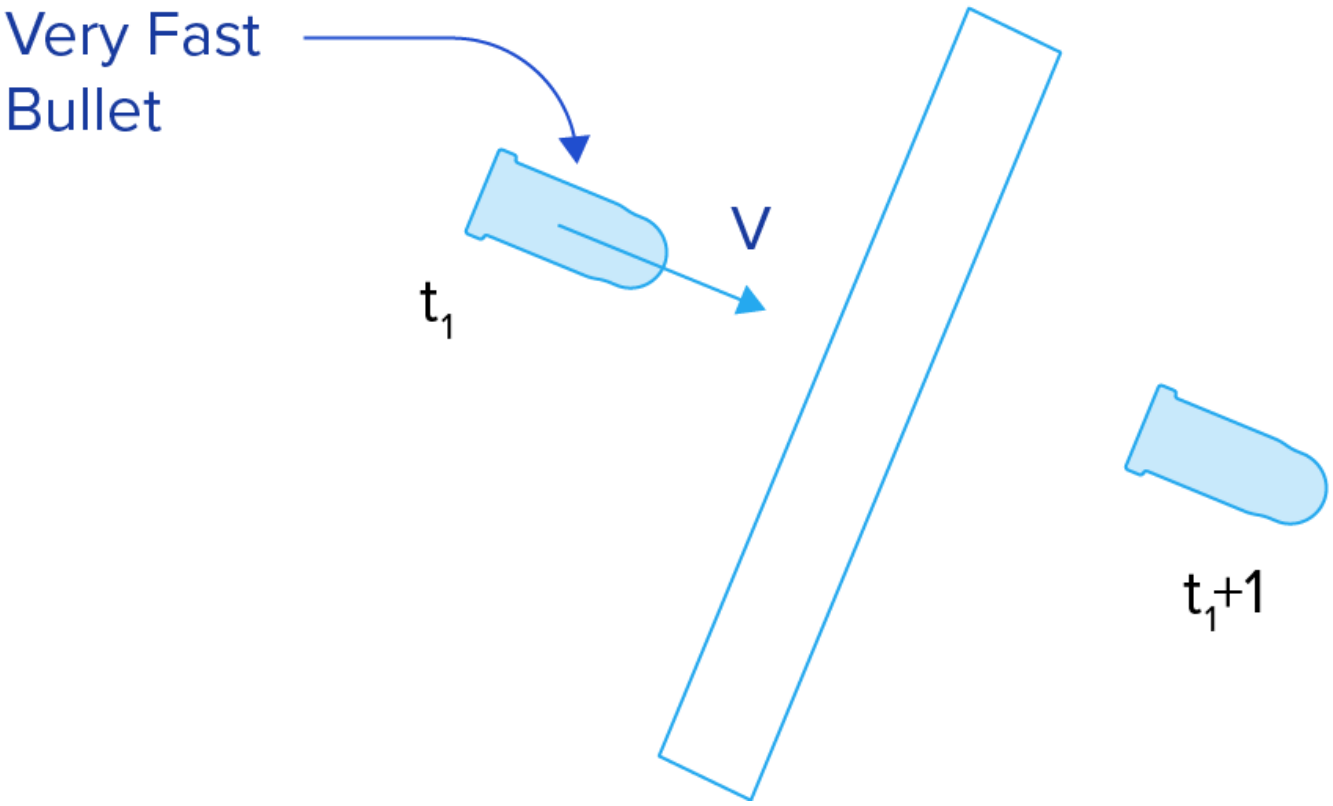


COLLISION DETECTION

Continuous Variant

Continuous Collision Detection

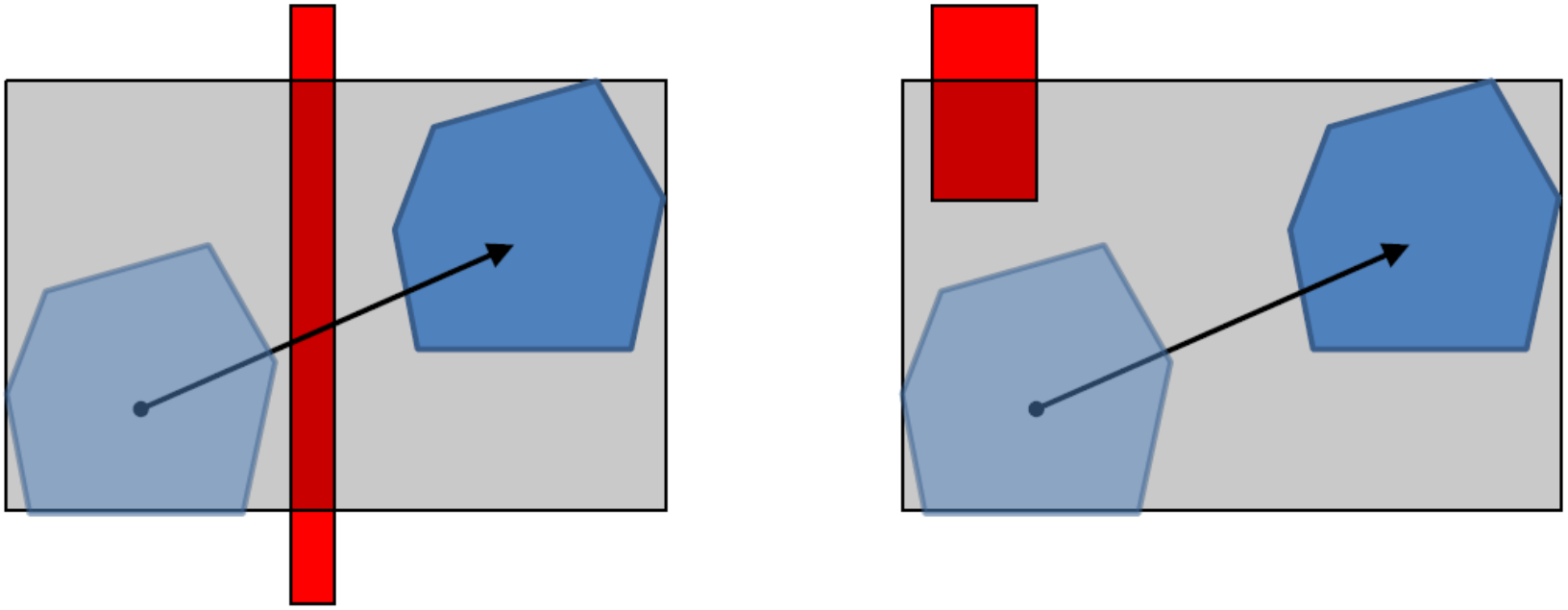
Tunnelling



Some collisions might not be detected for static snapshots, especially for fast moving objects. This issue is known as **tunneling**.

Continuous Collision Detection

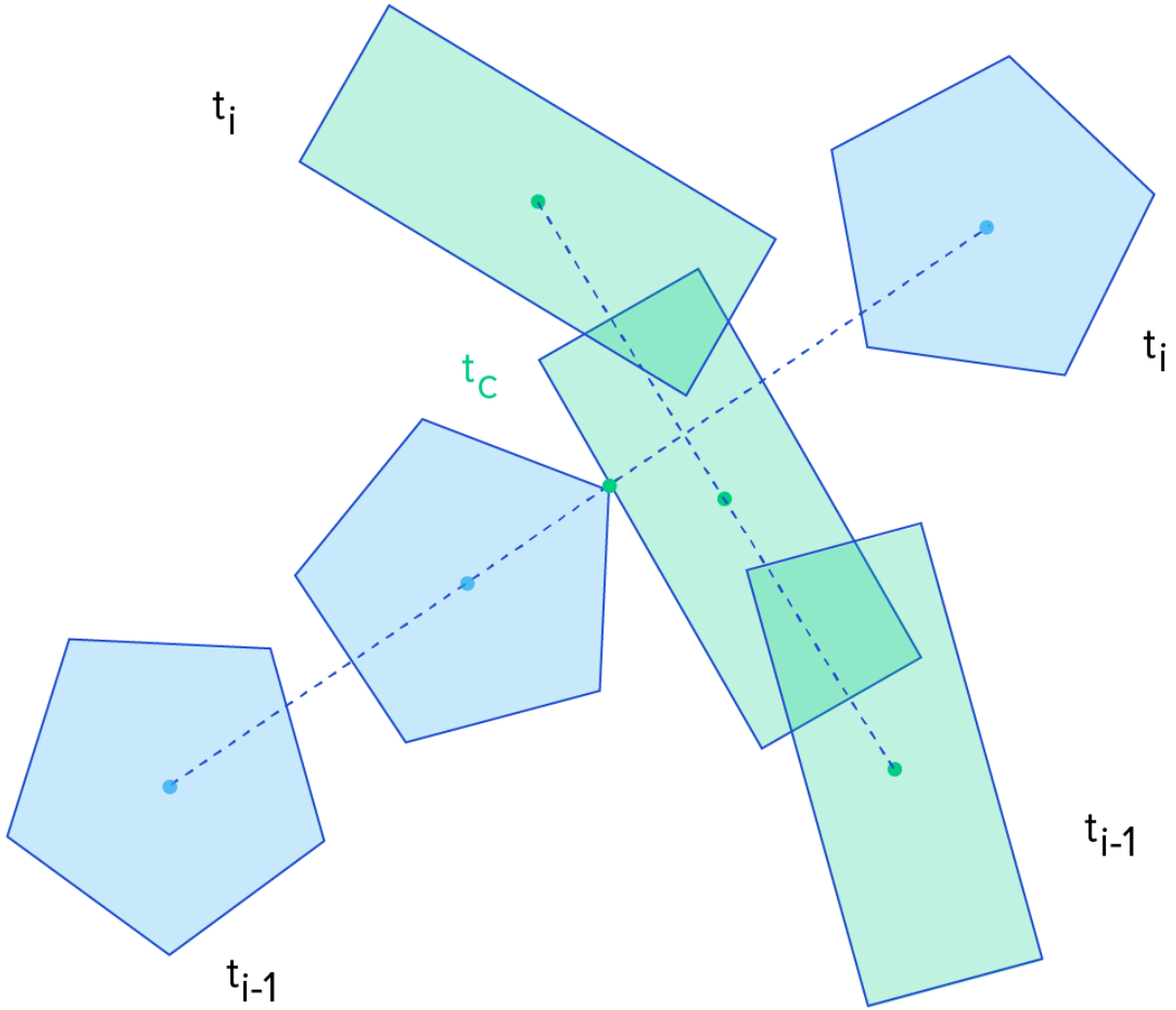
Broad phase



For broad phase, we can use movement bounds check. If they collide, there might be a collision.

Continuous Collision Detection

Narrow phase



That’s it for today!