Advanced Animation Programming

GPR-450

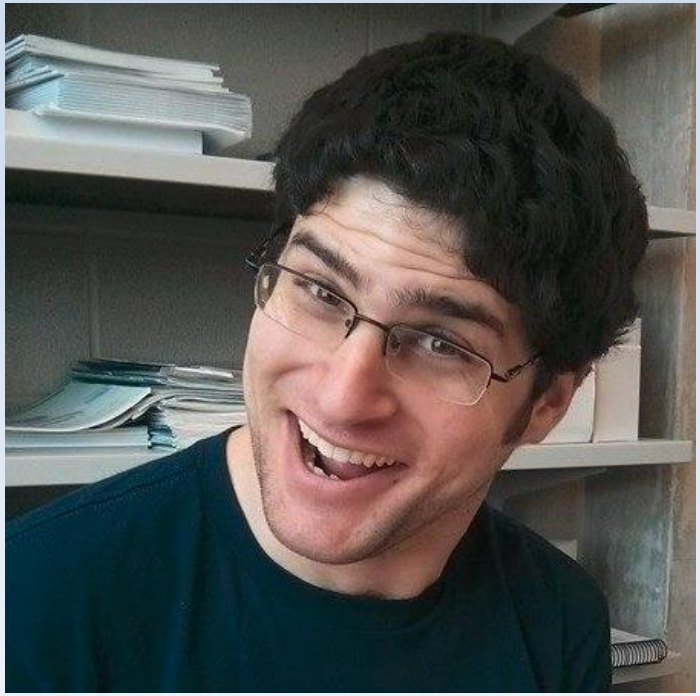
Daniel S. Buckstein

Course Introduction & Animation Programming Overview

Week 1

License

Introductions



- Course Instructor:

Dan Buckstein

M.Sc, Computer Science, UOIT

B.IT, ***Game Development*** & Entrepreneurship

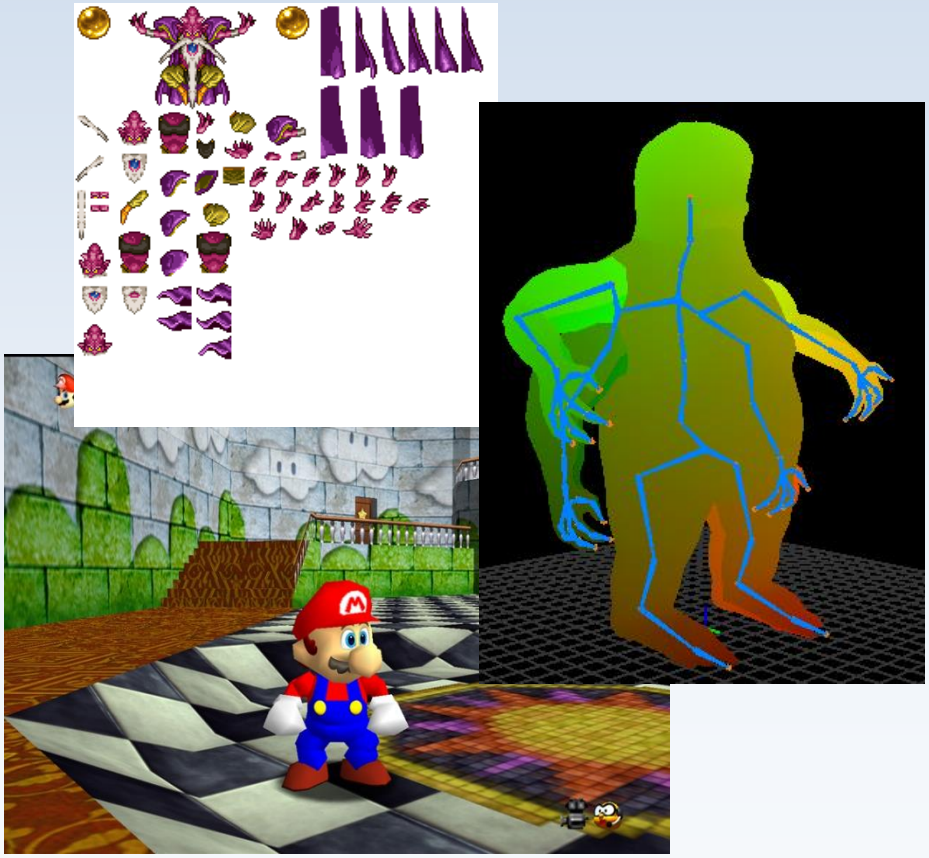
  - #uoitgamedev

- Favourite games: Dragon Quest I-VI, Super Mario 64, Banjo-Kazooie

ANIMATION PROGRAMMING

- What comes to mind?

- Sprites?
- Skeletal?
- Player control?
- All of the above?



What is this course?

- Intermediate to advanced animation programming
- Building fundamental interpolation and animation algorithms
- The code that brings the artists' creations to life in-game
- Building sandboxing/prototyping tools for game artists/animators (let them work!)

What is this course?

- What is this course ***to you***?
- Fundamentally two things:

1. ***Portfolio building***

   Projects are creative in nature, and will show employers what you can do in this domain

2. ***Engineering***

   Low-level & tools programming that applies all you have learned thus far in your courses

Why does this course exist?

- Breadth of expertise
  - Intermediate → advanced
- Learn to speak the other developer's language
- A past conversation with Chris Reese, director of Sony Bend studio:
  - Me: "What do you do when you find a good animation programmer?"
  - Chris: "Simple: we hire them."

How to succeed in this course

- Practice programming often
- Do work often and on time
- Attend all lectures and tutorials
- Attend office hours to clarify issues
- ***Do not procrastinate.***

- This is your education… make the best of it!

How to succeed in this course

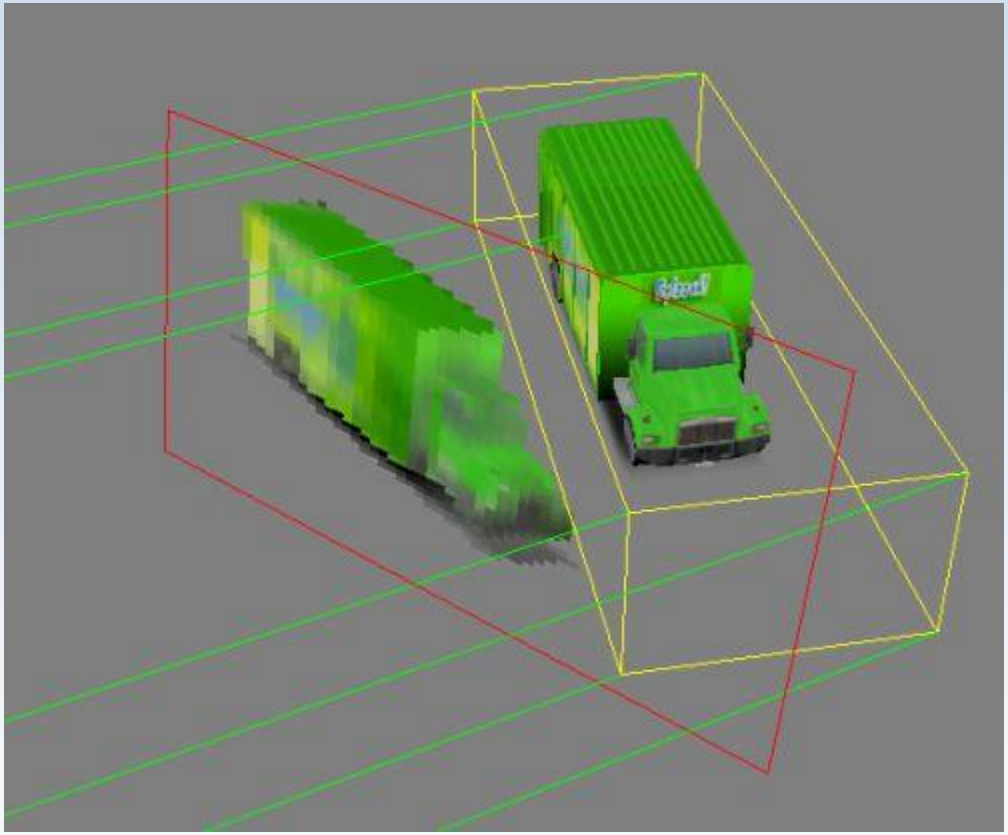
- Additional readings will be provided
- Do your own research to excel with the course content
- When in doubt…

  - …or just come find me

SYLLABUS REVIEW

- Course syllabus is posted on Canvas
- Find course link for ***GPR-450: Advanced Animation Programming***
- Syllabus is posted under the ‘Syllabus’ tab
- Other stuff posted under ‘Modules’

Impostor Syndrome



http://www.gamasutra.com/features/20060105/Figure5.jpg

Accessibility

- Again, feel free to approach me to discuss
- Works both ways…

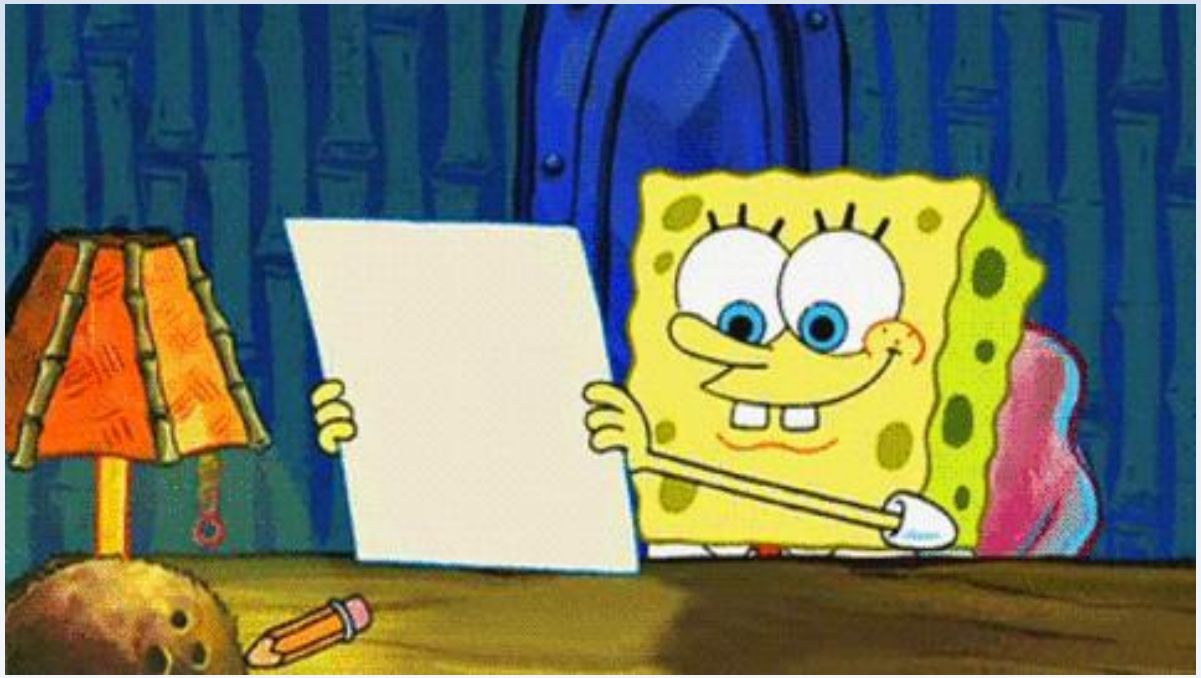
- Please sit closer to the front
- Please speak up
- Please do not mumble

Zero Tolerance for Plagiarism

Do not plagiarize.

Questions???

- Questions so far???



“Tools vs. Applications”

- Metaphor: what is this?

“Tools vs. Applications”

- Your survival in this course (and the rest of the program) relies partially on your ability to distinguish between ***tools*** and ***applications***
- ***TOOLS***: mathematical formulas, algorithms, theories, concepts, definitions…
- ***APPLICATIONS***: **use in your games!**

“Tools vs. Applications”

- Example: LERP

TOOL: The algorithm implemented in C/C++

```
vec3 lerp(vec3 v0, vec3 v1, float t)
{...}
```

APPLICATION: Move a character from A to B

```
myPos = lerp(posA, posB, posT);
```

APPLICATION: Colour blending

```
darkCyan = lerp(blue, green, 0.5);
```

“Tools vs. Applications”

- Math and programming go hand-in-hand!!!
- TOOLS vs. APPLICATIONS
- Algorithms are just mathematical formulas!!!
  - Tools
- Implementation of an algorithm is in code
  - Applications

“It’s Just Data”

- Course motto: ***“It’s just data.”***
- Remember this always!
- Algorithms can be used in many ways!
- Moral of the story: we are using algorithms to process ***data***
- Different purposes call for different applications of the same tools!!!

“It’s Just Data”

- Variables are just numbers
- Algorithms are just functions that take in and spit out variables

variable→ algorithm→ variable→ algorithm→…

```
float, int, vec2, vec3, mat4, frame,
keyframe, sequence, skeleton...
```

At the end of the day, it’s just stuff we process!

Frameworks!!!

- This year you should focus on building a solid ***framework***: a collection of tools (algorithms)
- Why bother?
- Do you want to implement your shader code every time you want to use it?
- Wouldn't you rather call a function or instantiate a class for any case or problem?
- TL;DR: simplify your life ☺

Frameworks!!!

- Introducing ***animal3D***: the minimal 3D animation framework

- Graphics
- Windowing
- Input… and more!

Use version control

- Recommended SCMs:
  - **Mercurial** (a.k.a. **Hg**), **TortoiseHg** for GUI
  - **Git**

:(

rip

- Course materials delivered using **Hg**

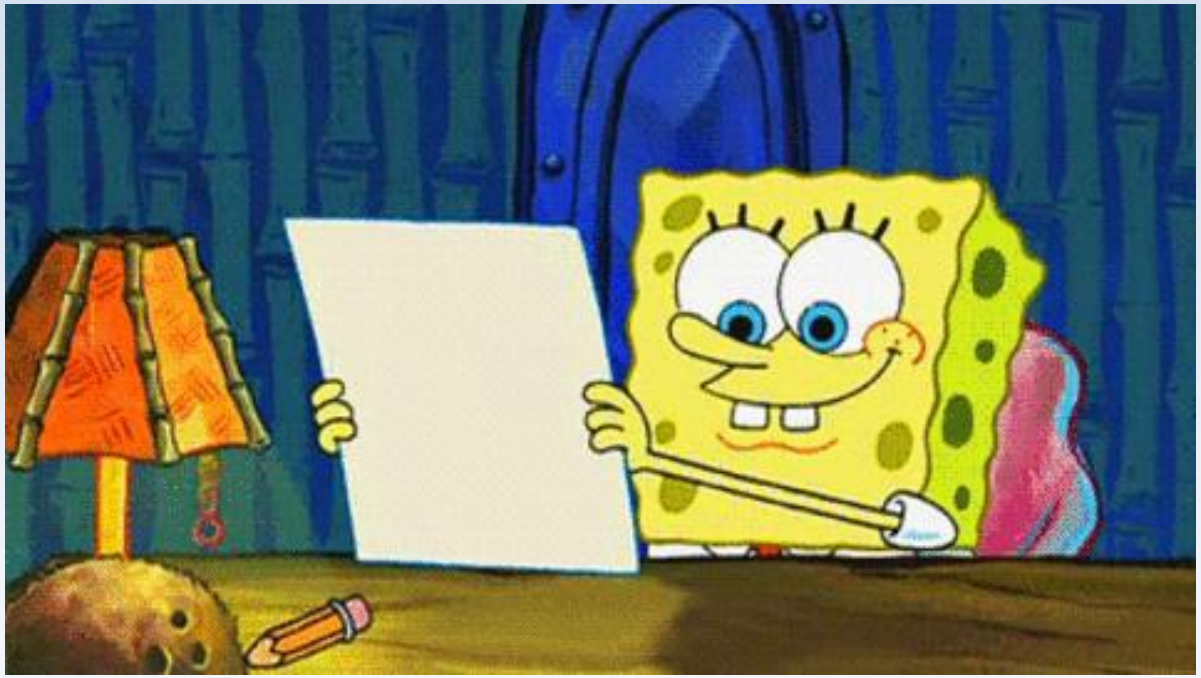
rip

Highly-Recommended Software

- DIY:
  - Visual Studio → programming IDE
  - Tortoise Hg (and plugin) → source control
  - p4merge → life-saving visual diff tool
  - Rapid Environment Editor → env. var. editor
  - FMOD Sound System → sound library & API
  - Everything Search → super fast file search
  - TeXstudio & MiKTeX → for fancy PDFs
- Dan's starter package:
  - animal3D: Minimal Animation → windowing, graphics
  - Developer SDKs → fun prerequisites

Questions???

- Questions so far???



Intro to Animation Programming

- A brief history.

Intro to Animation Programming

- A brief history… animated.

Doom footage

https://www.youtube.com/watch?v=8mEP4cflrd4

Banjo Tooie footage

https://www.youtube.com/watch?v=7RlB7mXcqlU

Call of Duty Modern Warfare footage

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

Dragon Quest 6 battle footage

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

Uncharted 4 animation breakdown

https://www.youtube.com/watch?v=2RkvTZdYqo8

https://www.youtube.com/watch?v=VtYrAvyKj4w

Uncharted 4 footage

https://www.youtube.com/watch?v=e5RLD4KJRMk

Sprite sheets galore

https://www.spriters-resource.com/

Intro to Animation Programming

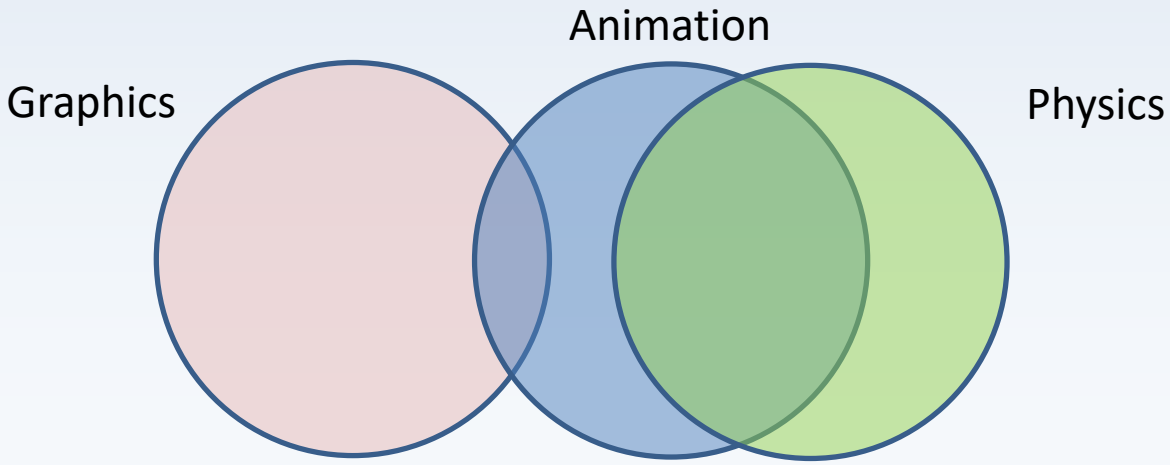
- The greatest challenge in animation programming???
- ***Time is your first dimension.***
- Everything can be thought of as a function of ***time***:

$$x = f(t)$$

- Generic **dependent variable** x changes as controlled **independent variable** t changes

Intro to Animation Programming

- Which discipline is animation programming's closest relative???
- ***Physics programming***
- Graphics is a close cousin



Intro to Animation Programming

- The key difference:
- ***Interpolation vs. Integration***
- All disciplines pivot around the concept of the ***derivative*** or rate of change: $x' = f'(t)$
- In physics, we *integrate* the derivative to go from one known state to the next unknown
- In animation, we *interpolate* between known states to emulate change over time

The end.

- Questions? Comments? Concerns?