

Hi

Welcome to the talk

My name is matt ditton

mditton@pandemicstudios.com.au
matt@polymonkey.com

And I ...

- have a degree in photography (from a long time ago)
- make computer games
- started in games as a modeler
- invented TY's boomerangs (hi kromans)
- went down the lead path (not for long)
- now I write code as well
- I'm teaching programming to artists (hi students)
- I'm currently the Senior Technical Artist for Team Alpha at Pandemic Studios OZ (hi pandas)

Today I'll be talking about
programming and artists

Specifically the idea of getting
artists to program.*

*this statement has many disclaimers

But first, a question for the
audience

It's the, "*who does what*" slide.

I'm doing this talk is for 2 reasons

- To show the benefit of Art Teams that include artists who know how to program.
- To introduce Processing.

And it'll be in 3 parts

1.

- Why should artists learn to code?
- What do artists gain?
- What does the team gain?

2.

- What kind of artists should be coding?
- What things should they code?

3.

- How do you teach them?

Part 1: Why?

I've got a few reasons.

1. Know your medium.

It makes you better.

Know your medium.

There was a time when artists had to make their own paint.

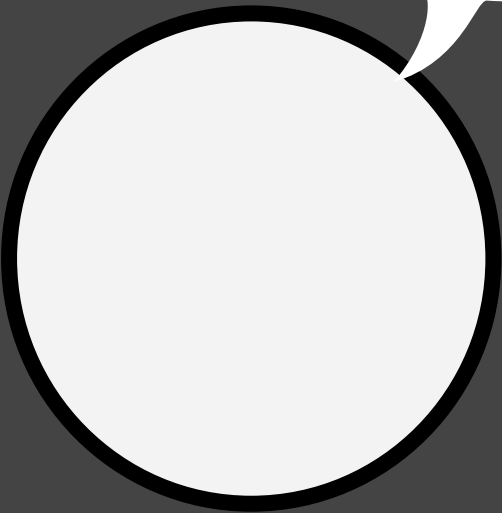
This practice informed the artist on the building blocks of their medium.

Your medium is computer games.

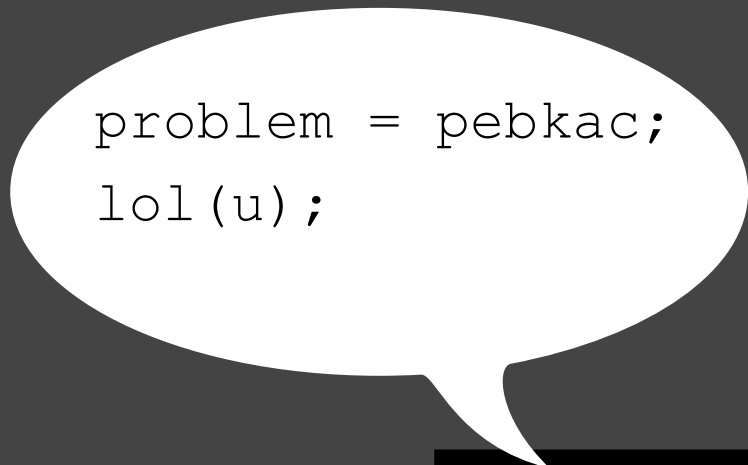
Shouldn't you know how they work?

2. Ignorance can lead to misunderstanding

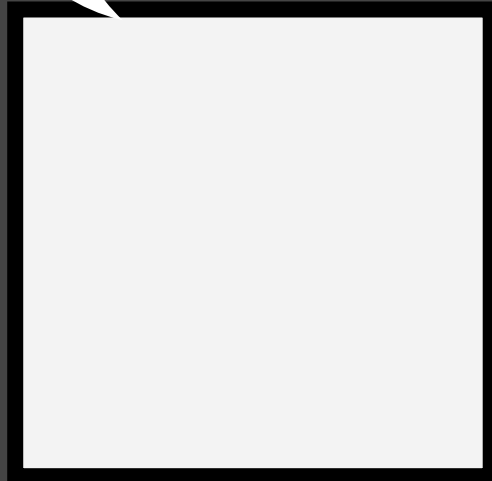
There is a story behind this.



why doesn't it
just work?



```
problem = pebkac;  
lol(u);
```



Often ignorance can get out of hand

For example, let's talk about the meaning of the word "value"

<http://dictionary.reference.com/browse/value>

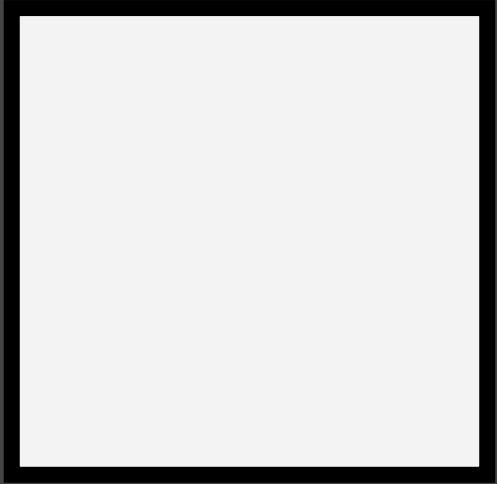
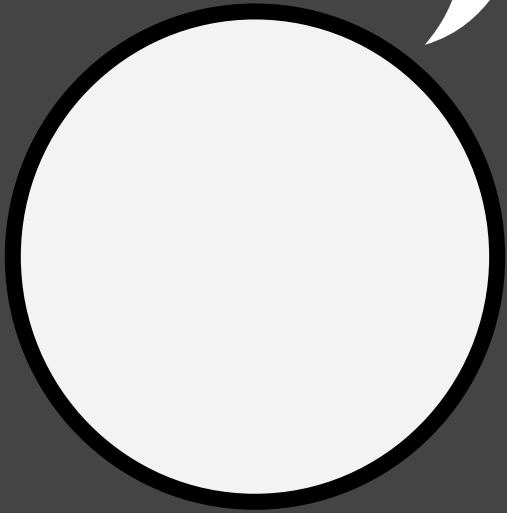
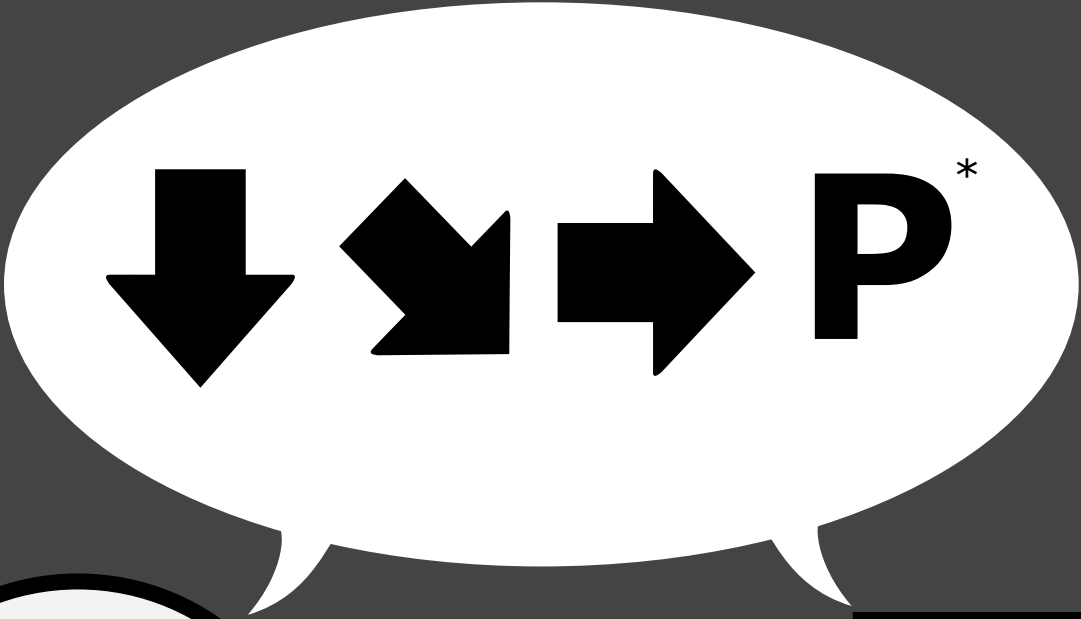
Mathematics.

- magnitude; quantity; number represented by a figure, symbol, or the like: the value of an angle; the value of x ; the value of a sum.

Fine Arts.

- degree of lightness or darkness in a color.
- the relation of light and shade in a painting, drawing, or the like.

2.a. When we understand each
other we get awesome



**hadōken*

3. We can own the data.

huh?

All that you make is data.

All we do every day is create.

Meshes, Textures, Animations, Shaders, Rigs, Levels,
Characters, Props, FX.

It all gets turned into data for the game.

Often what happens next is a total mystery to the art team.

This can lead to problems.

So owning the data...

When the Art team owns the data, it falls on them to ensure that it works, is clean, organized and efficient.

Better data is a win for the whole team.

- Programmers get to spend less time fixing problems.
- Designers are able to find resources that they need.
- People can rely on it working.
- Good data means less waste.
- Less waste lets you do more and draw more.
- Artists are able to make the game better.

4. We can own our tools

Back to that painters idea.

Programmers don't make art.

They don't spend the whole day working in ----*

They do not generate the volume of data we do.

They are fighting their own problems.

Would you want them making an art pipeline?

insert autodesk product here

The ideal world

In the ideal world there is an artist made and maintained pipeline.

It is made by the art team, and when it goes wrong (and it will) it is fixed by the art team.

There is a point of contact in the team for all matters relating to your work going into the game.

This person knows how to fix your problems because they built the system.

We'll be coming back to the ideal world a little later.

Part 2: Who and What?

Who should be learning to
program?

Should all artists learn?

Yes.

Life is easier when you know how it works.

Frees you from just pushing buttons.

Frees you from waiting for the feature.

You can do some pretty cool stuff.

No.

Takes them off the path of their chosen specialty

Isn't it hard enough?

"But I'm an Artiste"

I'm scared.

Too much typing, not enough clicking

Well let's be pragmatic.

Start with the Tech Artists

What makes a tech artist?

- Package Specialist
- Riggers
- Tech Animators
- Fx Artist
- Lighting Specialists
- Scripters
- Vehicle Specialists
- Anal retentive artists
- An artist who once spoke to a programmer and their eyes didn't glaze over.

What should they write?

Why not start big?

How about the exporter?

The exporter is the starting point for getting art into the game.

There is no better way to ensure that the system is understood than to have an artist build the exporter.

How can this be done?

I'm glad you asked.

Exporter writing tips.

for the tech artist

How an Artist can make an exporter

Open the format

- We use Collada and it has saved us a MASSIVE amount of money and time.
- It's native to XSI, the libraries are open and the crosswalk exporter is fully script-able.
- You don't have to re-invent the wheel.
- And you get an importer for free :-)

How an Artist can make an exporter

Pick an open language, learn to use an IDE

- Every 3D package has a scripting language.
- Learn skills that you can transfer.
- Closed languages don't get you very far.
- I'd go with Python or JavaScript.
- I use eclipse. It's free.
- If you are writing scripts in notepad. Stop hurting yourself.

How an Artist can make an exporter

Script away the repetitive bits

- Selecting the right stuff in the scene.
- Getting the file name right.
- Saving to the right folder.
- Checking the file out of source control, or marking a new file for add.
- Copying tga's to the right location.
- The above can all be done through a minimum of user input.
- Stop forcing people to waste time.

How an Artist can make an exporter

Provide useful feedback when something goes wrong

- Alerts in the first person go over well. "I'm Sorry" goes a long way.
- Tell people how to fix the problem they have hit.
- When it all goes wrong tell people to come and get you.

How an Artist can make an exporter

Start small, build trust.

- You'll never get it right the first time.
- Only iteration can build a quality tool.
- Don't go silent when writing anything.
- Talk to the people who are using it. (sit next to them)
- If you stuff up, drop everything and fix it.
- NEVER NEVER NEVER let the team lose trust in the tool.
- The worst thing that can happen is if bugs cause people to stop using it on suspicion.

How an Artist can make an exporter

Make it clean. Make it simple.

- There are countless theories about the "one button" export.
- There is no need to make it huge and over the top.
- There is no need to blast people with information.
- There is no need to confuse people.
- Do more with less input.
- Just keep it simple.

But there is plenty more to
write

We need these too...

- Shader front end
- Material system helpers
- Auto skinning tools
- Auto Lodding tools
- Collision creation tools
- Source control integration
- Scene cleaning
- Scene setup
- Animation helpers
- Morph controllers
- Meta Data exporting
- File indexing (this is quite useful)
- Bulk data creator

3. How do you teach them?

What not to do...

- Don't point to a help file and say "*best of luck*".
- Don't start with C++
- Don't start with Visual Studio.
- Don't make fun of them.
- Don't scare them.
- Don't assume it's common knowledge.
- Don't flood people with information.

So... Teach Processing.

It's worked for me.

Why processing?

- People learn best when they want to learn.
- Artists learn when they are creating.
- It offers constant visual feedback.
- It's non threatening
- It's simple but expandable.
- Industry standard syntax (It's Java with the crap gone).
- It teaches coding conventions.
- There is fantastic help.
- Can export to a browser.
- Can make PC/MAC/Linux apps with one button.
- It's the best sandbox you'll ever play in.
- And it's free.

Let's have a look.

For the viewers at home...

At this point in the talk, I've given a run through of processing and shown examples of how I've used it in production. But you should have a look at

<http://www.processing.org>

Download it and have a play. That's how we learn.

If you get stuck feel free to email me.

Examples - collections

The processing exhibition page

<http://www.processing.org/exhibition/>

Processing Blogs

<http://www.processingblogs.org>

flickr and vimeo

<http://www.flickr.com/groups/processing/pool/>

<http://www.vimeo.com/tag:processing>

google "built with processing"

Examples - people

Ben Fry (the guy who wrote it)

<http://acg.media.mit.edu/people/fry/>

Casey Reas (the other guy who wrote it)

<http://reas.com/>

Jared Tarbell

<http://www.complexification.net/>

Robert Hodgkin (if you want to look at pretty things, start here)

<http://www.flight404.com/blog/>

Jonathan Harris

<http://www.number27.org/>

Examples - projects

Radiohead - "house of cards"

<http://code.google.com/creative/radiohead/>

White Glove Tracking

<http://www.whiteglovetracking.com/>

10000 cents

<http://www.tenthousandcents.com/top.html>

The End.

matt@polymonkey.com
mditton@pandemicstudios.com.au