Tasveer Singh

Good Evening Lady/Ladies and Gentlemen
My name is Tasveer Singh...

Taz

But you can call me Taz

And I’m...

zenapsis™

the President and Co-founder of zenapsis

I love Rails

and I would like to say that I love Rails.
It's great and amazing and awesome. But I've been doing it for about 5 years now, so I'm getting a little bored.

Thankfully, Rails introduced me to Ruby and I've been sneaking out to see her for a while now (Shh don't tell the gf)

I thought I'd share my... evenings... with Ruby with you to get away from the Rails centric talks of the Brigade nights.

So I'll most likely be giving a talk next month on Ruby on Android so you can look forward to that.

But for now, let's get into video games with...

Gosu

What is Gosu?

- Gosu is a 2D game development library for the Ruby and C++ programming languages, available for Mac OS X, Windows and Linux. The C++ version is also available for iPad, iPhone and iPod Touch.
- Ruby bindings to C++ code through SWIG

First of all, What is Gosu?

From the Gosu website: ... (first point)...

Basically, it's... (second point)...

SWIG - Simplified Wrapper and Interface Generator
Connects C/C++ code with high-level languages
Perl/PHP/Python/Tcl/Ruby

Gosu is used in Video Game Competitions for Rapid Game Development because of Ruby
And it's fast thanks to it's C++ bindings

Getting Started

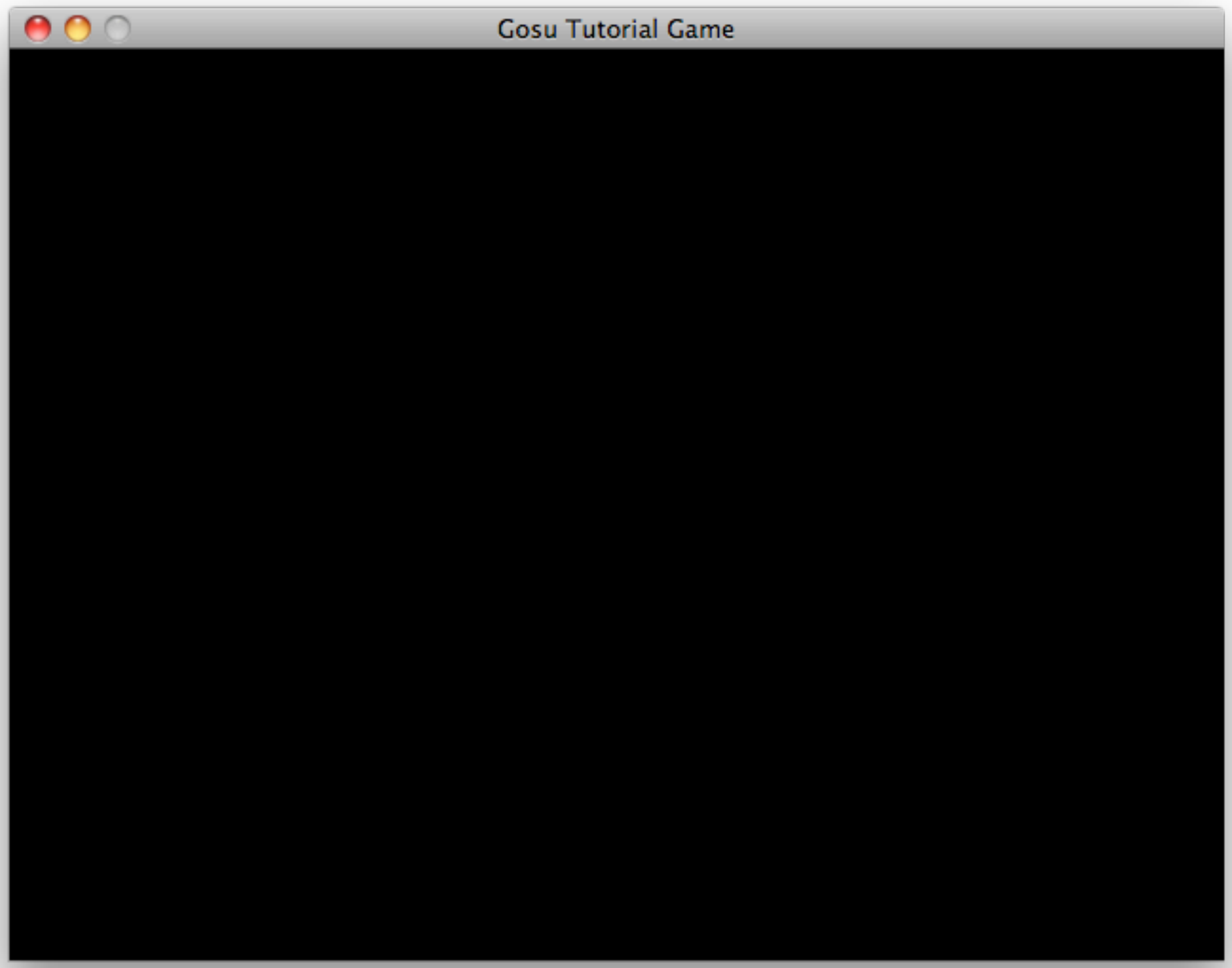
```ruby
  require "gosu"

class GameWindow < Gosu::Window
  def initialize
    super 640, 480, false
    self.caption = "Gosu Tutorial Game"
  end

  def update
  end

  def draw
  end
end
GameWindow.new.show
```

Install with
gem install gosu
Linux users have to install extra libraries via apt
Look at Gosu Wiki, linked to in references

Taken from the Ruby Tutorial on the Gosu Wiki

Run through Code
super 640, 480, false
false means don't run in fullscreen mode

Runs update method, then draw method, then update method, and so forth



Gives you something like this

If you are familiar with C++ video game design, you know that to create a window you need about 100 lines of code.

Here we've done it with just 15

Ok... What else?

- Classes
  - Gosu
  - Color
  - Font
  - GLTexInfo
  - Image
  - Sample
  - SampleInstance
  - Song
  - Window
- Extensions to Numeric
- Notable Functionality
  - Gosu::Window#gl

Classes
Gosu includes shortcuts for I/O events, such as Key Codes, Mouse Events, etc.
Math functions, such as determining angles, distances, and movement in a direction

Colour maps Hex colours with their Alpha and RGB or HSB channels to Gosu Colours

Font will accept a Font and draw Text for that font

GLTexInfo provides extra information for certain Images to be used directly with OpenGL
More on this later

Image handles... well, images
Can create from Text or from File and can draw that to the window

Sample is a short sound clip which you can only play

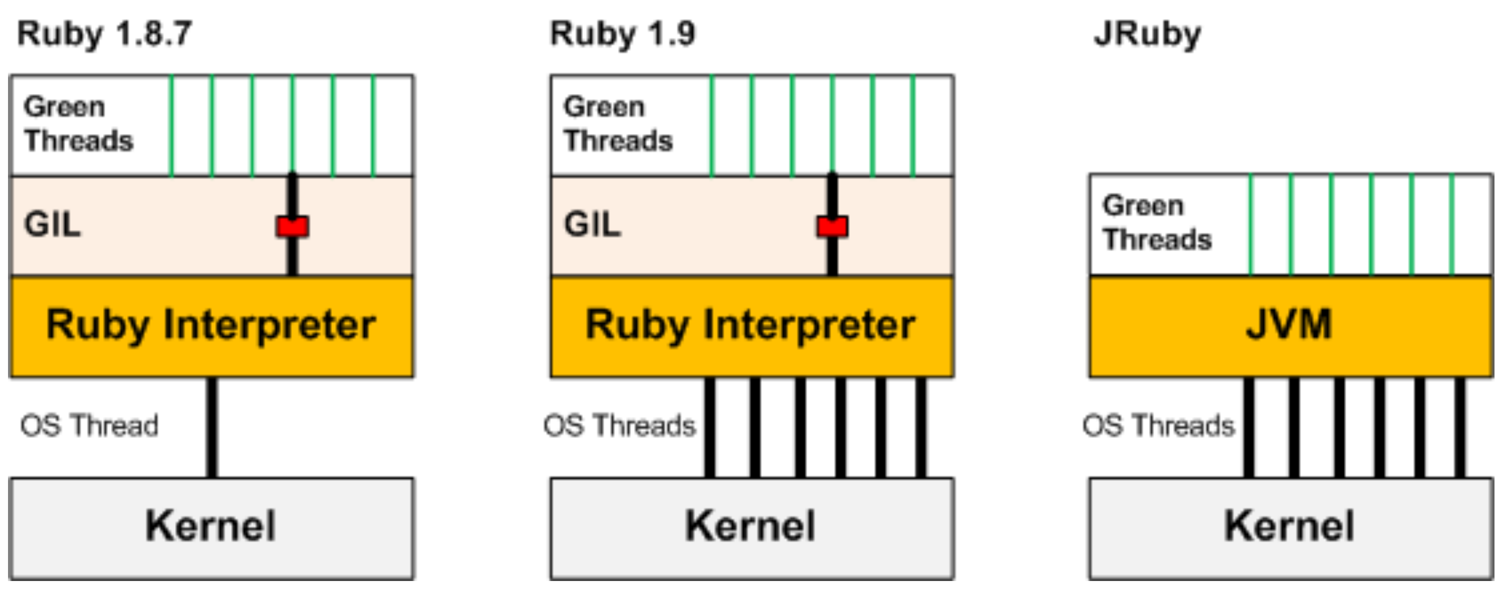
SampleInstance provides extra functionality to Sample
pause, playing?, resume, stop

Song is for background music
Only one Song can be played at a time
Has functionality to determine which song is playing, so you can change the song if desired

I showed you Window in the last slide
This is the main Class of your Gosu game
All the coordinates of the game are in relation to the Window, as it is in a browser
You should have only one instance of this Class
Window does everything from drawing lines, quads, and triangles to rotating, scaling, and translating drawings

Caveats

- Single Threaded
  - Global Interpreter Lock



There are caveats to Gosu, not necessarily with Gosu but with Ruby

Ruby is Single Threaded as it is not thread-safe
Implemented by the Global Interpreter Lock

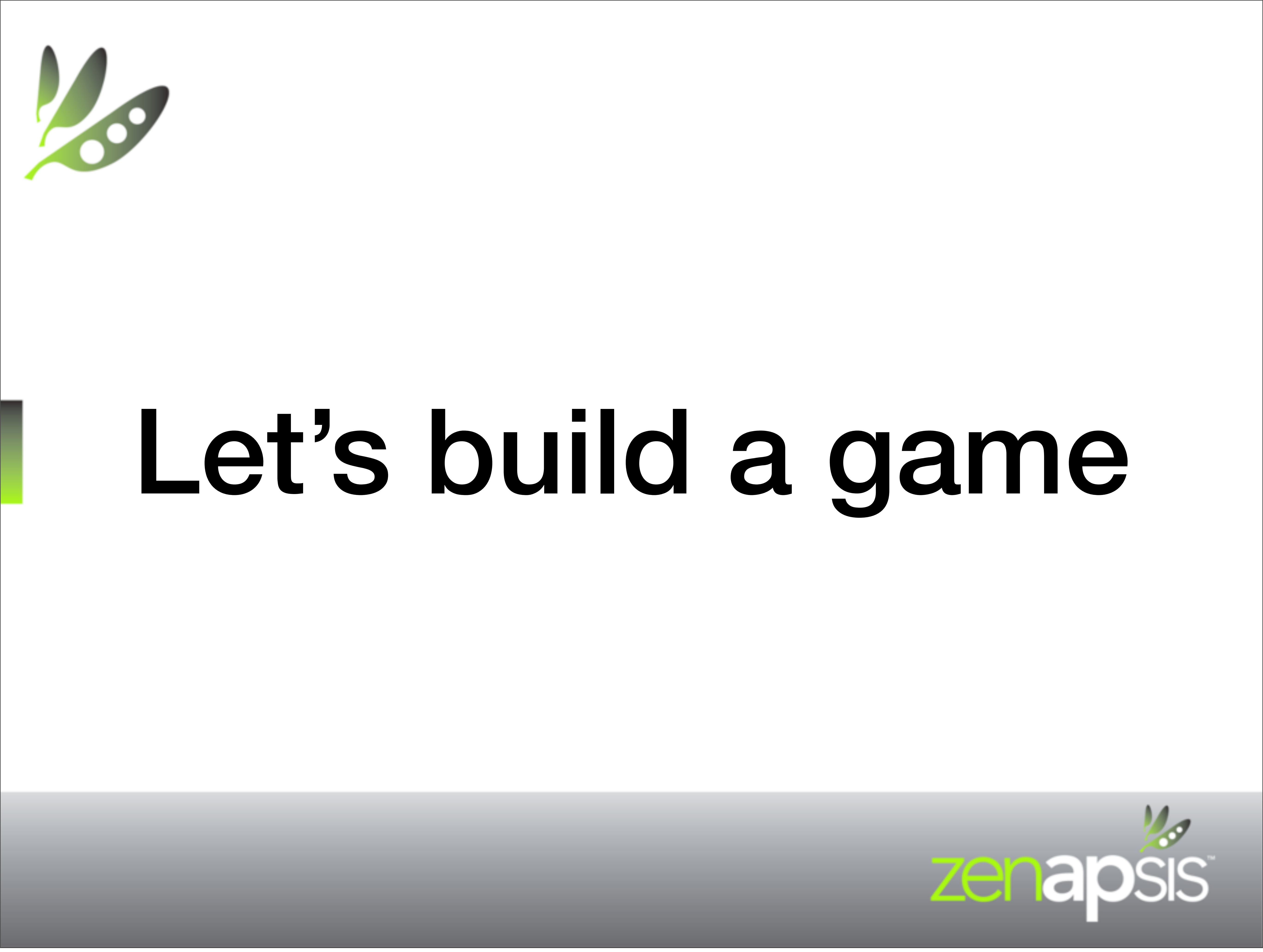
Can be a disadvantage for games which require heavy processing

Use the C++ Library or JRuby
I haven't used either with Gosu, so you're on your own

Diagram taken from Illya Grigorik's blog, CTO of PostRank who recently was acquired by Google.

Explain the diagram

Ruby 1.8 has a single OS thread between Kernel and Interpreter
Ruby 1.9 bottleneck is the GIL which allows only one thread to execute at a time
JRuby compiles Ruby to bytecode and then executes it on the JVM
Ruby threads are mapped directly to OS threads, with no GIL to limit them



Let’s build a game

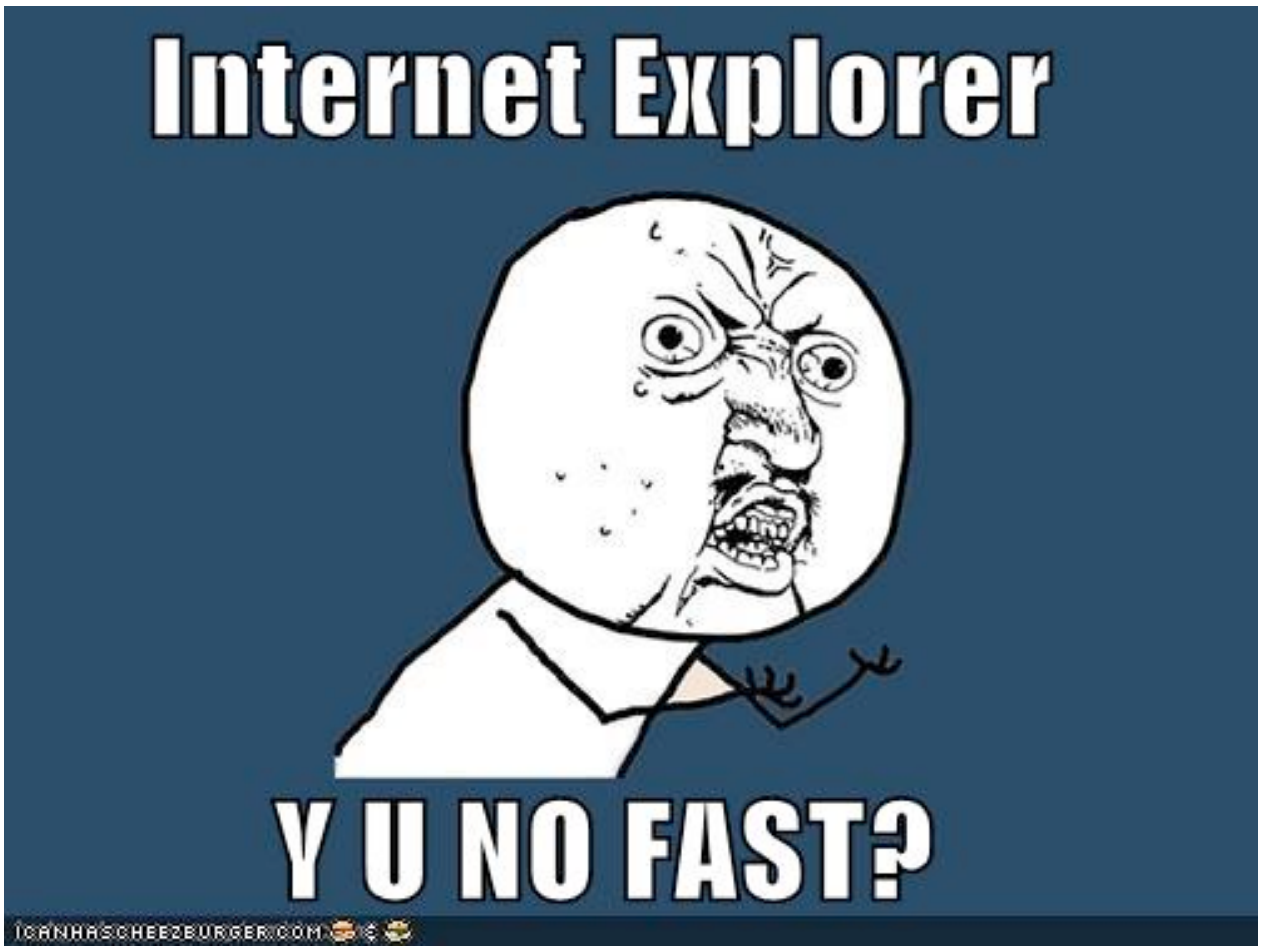
zenapsis

This was a tricky one for me... What game do I build?

A side scroller like Mario? Where you’re hopped up on ‘shrooms half the time?
Tron clone?
Tetris?
Pong?

Well actually let’s back it up... First I need an enemy
So as a web developer.... What is my biggest enemy? Anyone have any ideas?

... Yes! Internet Explorer 6!



I threw together some images that express my rage for IE6

Now that we have our enemy... What’s our hero?

Hmmm... as a web developer once again... What do I love?

Apple Crack.

As you can see I have a Mac, iPhone, iPad at home
I am addicted to Apple Crack

Steve Jobs day yesterday just continued to feed my craving

The Game

- You are a Web Developer
- Enemy is IE6
- Hero is Apple
- Game is...?

- Tower Defense
  - Shoot IE6 Bugs
  - Apple Product Towers
  - Stop IE6 Bugs from getting to End User

zenapsis™

Web Developer

Enemy is IE6

Hero is Apple

Game is....

Tower Defense!
Talk about Tower Defense games
Enemies follow Road
Towers shoot down Enemies

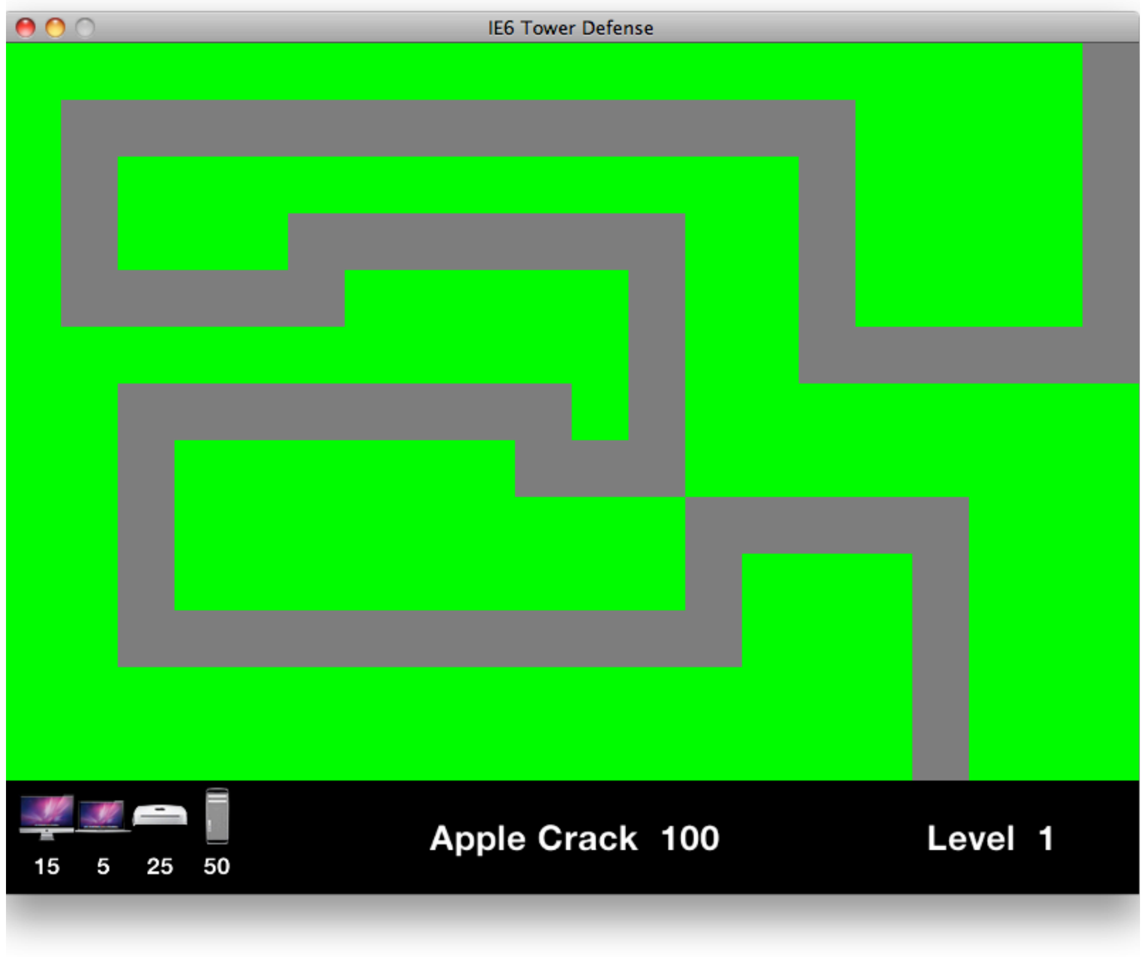
IE6 Bugs
Alpha Transparency
Hover State
Floating Elements
Rounded Corners

Apple Towers
iMac
MacBook Pro
Mac Pro
Mac Mini

You are a web developer
Stop IE6 Bugs from getting to the end user’s browser
Make money as you do this
Spend money on Apple Crack

Items to Consider

- Winning and Losing
- Getting Points
- Enemy AI
- Collisions



Things to consider when making your game

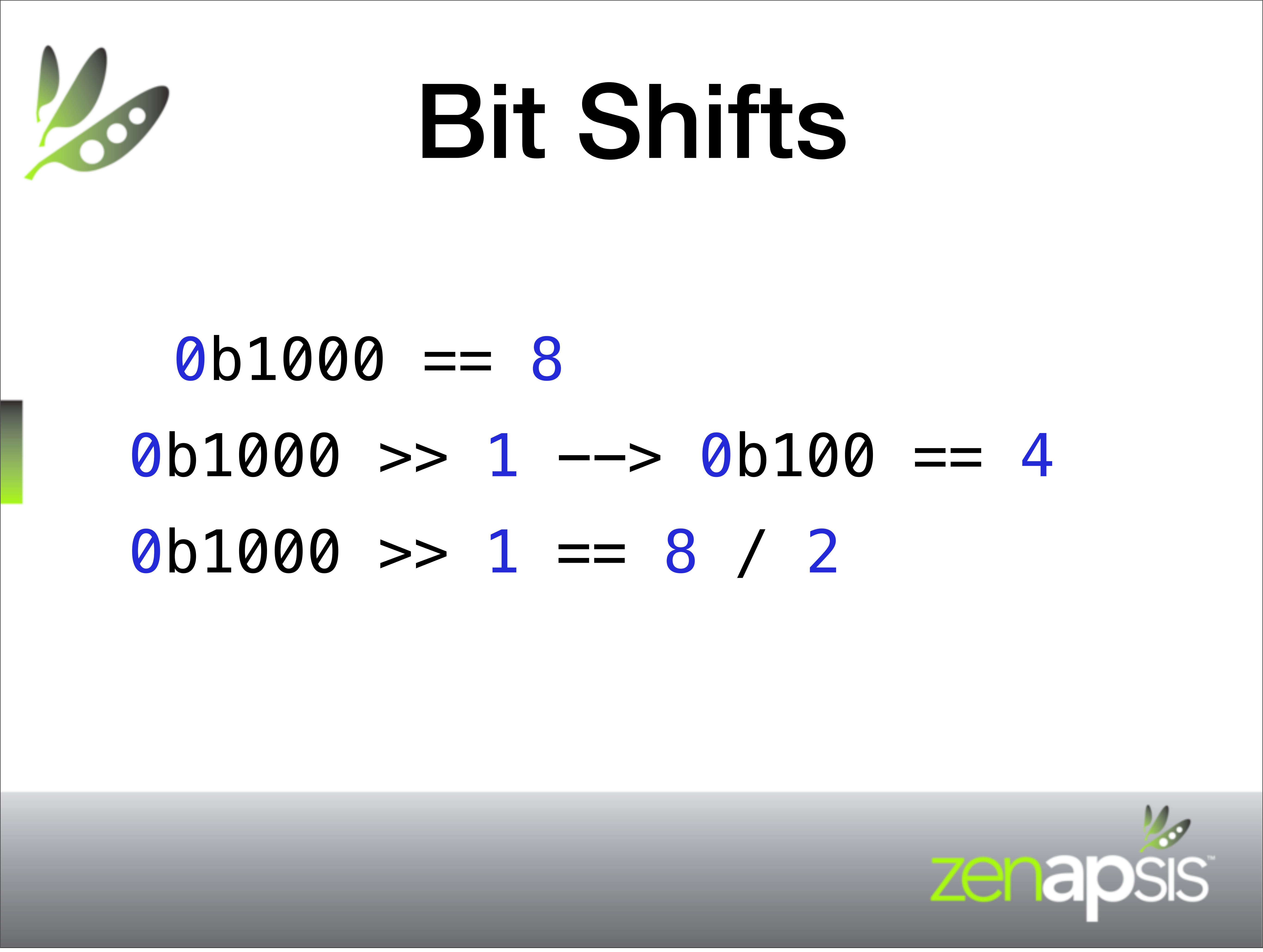
Can someone Win? If so, how do they?
How does someone Lose?

Getting points by killing enemies? Dodging balls? Taking the least swings in golf?

Enemy AI as in what does the enemy do.
Please note that the term AI here refers to Weak AI, as in the enemy performs a set number of tasks.
According to Wikipedia, "an artificial intelligence system not intended to match or exceed the capabilities of human beings, as opposed to Strong AI, which is."
Basically, do they shoot back? Do they run around? Do they know where you are?
Think of the AI needed for a game like Pacman. The ghosts need to chase you down until you eat a power pellet (the oranges), at which point they need to run away

Collisions are a big part of any video game
How do items collide/die?
Fortunately we are only dealing with 2D collisions and do not have to take into account a third dimension

Go over these points in regards to a Tower Defence game



Bit Shifts

```
  0b1000 == 8
0b1000 >> 1 --> 0b100 == 4
0b1000 >> 1 == 8 / 2
```

zenapsis

Bit shifts are used to shift bits either left or right

Since binary is of base 2, shifting bits left or right will either double or halve the value depending on which way you shift it

This is a common practice in video game design when either multiplying or dividing by a power of two, as it only requires a single CPU instruction to do so

When I was making my game, I followed this principle of video game design and thought I'd put a benchmark together to show you how much more efficient it is

Benchmark

```
  require "benchmark"

Benchmark.bm(5) do |x|
    x.report("div")
{ 100_000_000.times {|i| i / 4 } }
    x.report("shift")
{ 100_000_000.times {|i| i >> 2 } }
end
```

So this is my benchmark

Basically divide numbers up to 100 thousand by 4
Then bit shift numbers up to 100 thousand by 2

Result

```
             user     system      total        real
div      6.920000   0.000000   6.920000 (  6.916998)
shift    7.630000   0.010000   7.640000 (  7.636034)
```

This was the result...

Wait what?!

Bit shifting took almost a second longer??

Why?

I took a look at the Ruby source code written in C

Basically, to shift right, it has to check the size of the Fixnum to see what C data structure to use
Then check if you're moving right by a negative amount, which runs the left shift function
If it hasn't returned by now, then only call the shift right function

While for fixed point division, all Ruby does is check the size of the divisor (in this case 4)
Since it's a small number, simply call the division algorithm and return the result

It's a very small difference, but it's still a difference and goes against that core video game performance technique

What I'm trying to say here is: With higher level languages, just use the right tool for the job. The underlying C code is generally smart enough to be optimized for what you're most likely trying to do. Obviously run your own benchmarks as I'm sure this doesn't apply everywhere

Demo

Demo

Questions?

Follow Me
@tazsingh

Questions?

Ruby 4 Kids
Game Development in Spare Time

Thank you

References

- https://github.com/jlnr/gosu
    - https://github.com/jlnr/gosu/wiki
- http://swig.org/
- http://www.libgosu.org/
    - http://www.libgosu.org/rdoc/files/README_txt.html
- http://www.igvita.com/2008/11/13/concurrency-is-a-myth-in-ruby/
- http://en.wikipedia.org/wiki/Weak_AI
- http://en.wikipedia.org/wiki/Strong_AI
- https://github.com/ruby/ruby
- http://ruby4kids.com/