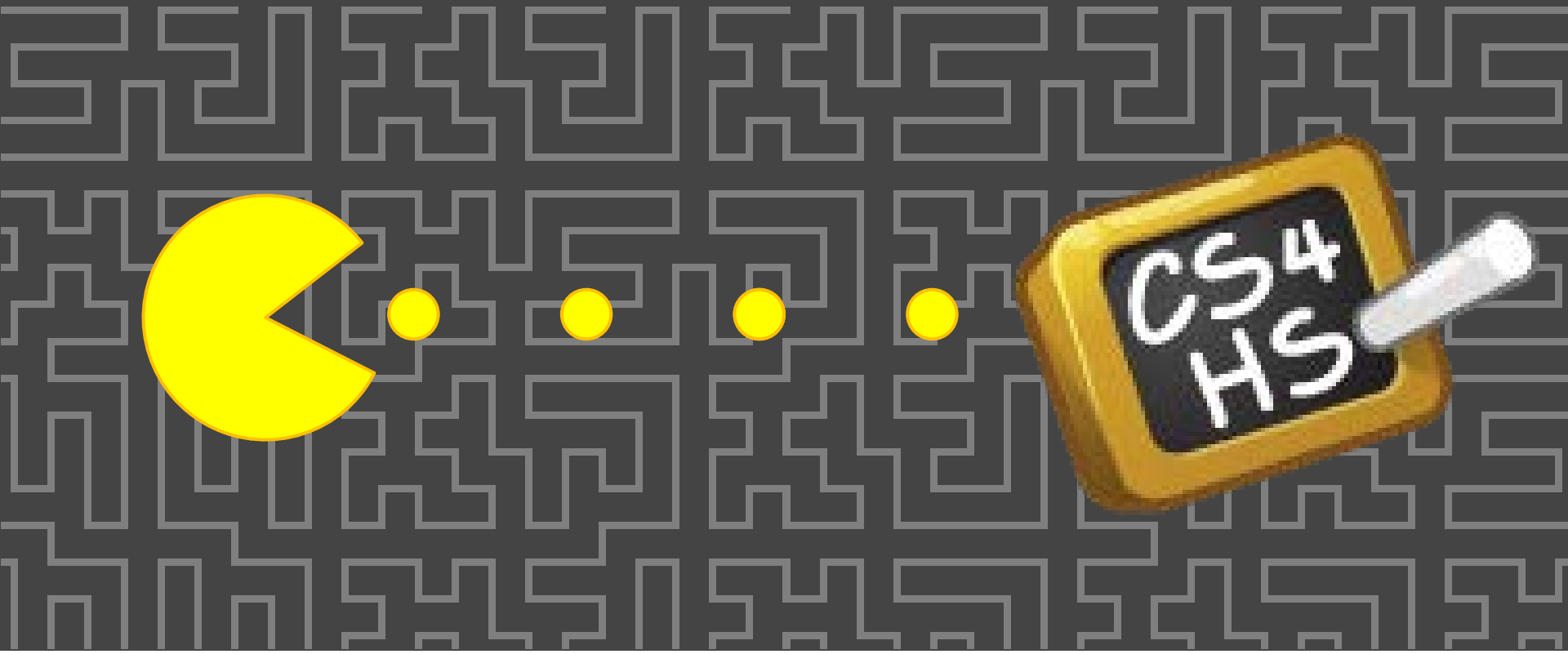
MOBILE PROGRAMMING



Making Computer Science Fun

Content

- Environments and Programming Languages
  - Pre-requisite knowledge
- Mobile Computing
  - Sensors
  - Screen Size
- Google Play
- Android Architecture
- Code Demonstrations
- Example Student Work

Why Mobile Programming?

- Students can create something that is tangible – something that they have developed themselves
- Helps students understand mathematics in a wider context
- Covers a whole range of real world skills: problem solving, critical thinking, logic etc.
- It is a lot of fun!
- Students can interact with and develop code for a lot of new devices and sensors
- Accessible with environments like MIT App Inventor

Languages

- Apple iOS
  - Objective C
  - Pointers
  - Object Oriented
  - Automatic Reference Counting
  - Need an Apple Computer and a Developer Account
  - Need to know about design patterns such as Model View Controller
  - Too complicated for a first time programmer

Languages

- Windows Phone
  - Visual Studio based IDE
  - Free Visual Studio Express IDE Student Edition
    - Missing advanced debug features: conditional break points, debug control panel.
    - No extensions
    - Actually quite a lot of features for a free version
  - Object Oriented
  - Need to know C#

Languages

- Android
  - Java bytecode compiled into DEX bytecode
  - Need to setup both Eclipse and Java
    - Lots of situations where the setup will fail… mismatch of IDE mismatch of Java versions etc.
  - Object Oriented
  - Partially message driven
  - Still too complicated for a new programmer
  - AppInventor makes it a lot a lot easier to do fun things quickly

Languages

- HTML 5 Canvas
  - Not a native application
  - Easy to setup a free syntax highlighting editor
  - Need to pay for an editor that does intellisense properly
  - Code can run on any browser that supports HTML 5 including mobile browsers such as Safari on iOS and Chrome on Android
  - Comes with all the downsides of writing code in JavaScript
  - Students can do *cool and interesting* things quickly!

Pre-requisite Knowledge

- Algebra
- Cartesian Geometry
- Trigonometry
- Vectors and Matrices

Mobile Computing

- Screen Size
- Computer Architecture
- Network Connections
- Input / Output

Screen Sizes

- Small variable in size
- Some devices have limited hardware acceleration

Computer Architecture

- Processors
  - Most seem to be ARM based, clock speed is mostly on the low end, but highly variable.
  - Some processors feature built in GPUs and multiple cores (see Nvidia Tegra)
- Memory
  - Limited: mainly 256 MB – 1024 MB
- Storage
  - 20+ GB is common

Network Connections

- Network connections
  - Low bandwidth
  - High latency (poor RTTs / pings)
  - Poor connectivity
  - Dependent on how busy the network is
- Devices generally support 3G/4G, Wi-Fi, GSM etc. and switch between different networks often
- Need to make sure applications can tolerate transient network / internet connectivity and dropouts

Input / Output

- Keyboard still exists
  - But might be a virtual keyboard
  - Might have a small hardware keyboard
  - Upside: writing a long novel is going to be hard
- Mouse
  - Most devices are touch screen, which is similar to a mouse
  - Lots of devices have extra navigation buttons like a home screen button or a back button

Input / Output

- Sound
  - Speech recognition
- GPS
  - Typically interfaces to Google's Maps API
- Vibrate
- Accelerometer
- Cameras
- Compass

Mobile Market

- Applications are a lot cheaper or free
- A lot easier to distribute and easier to develop your own application
- Developing a successful desktop application is probably going to be difficult unless you work for Microsoft or Adobe
- Any one can create a mobile phone application, upload their app to an online store and start making money

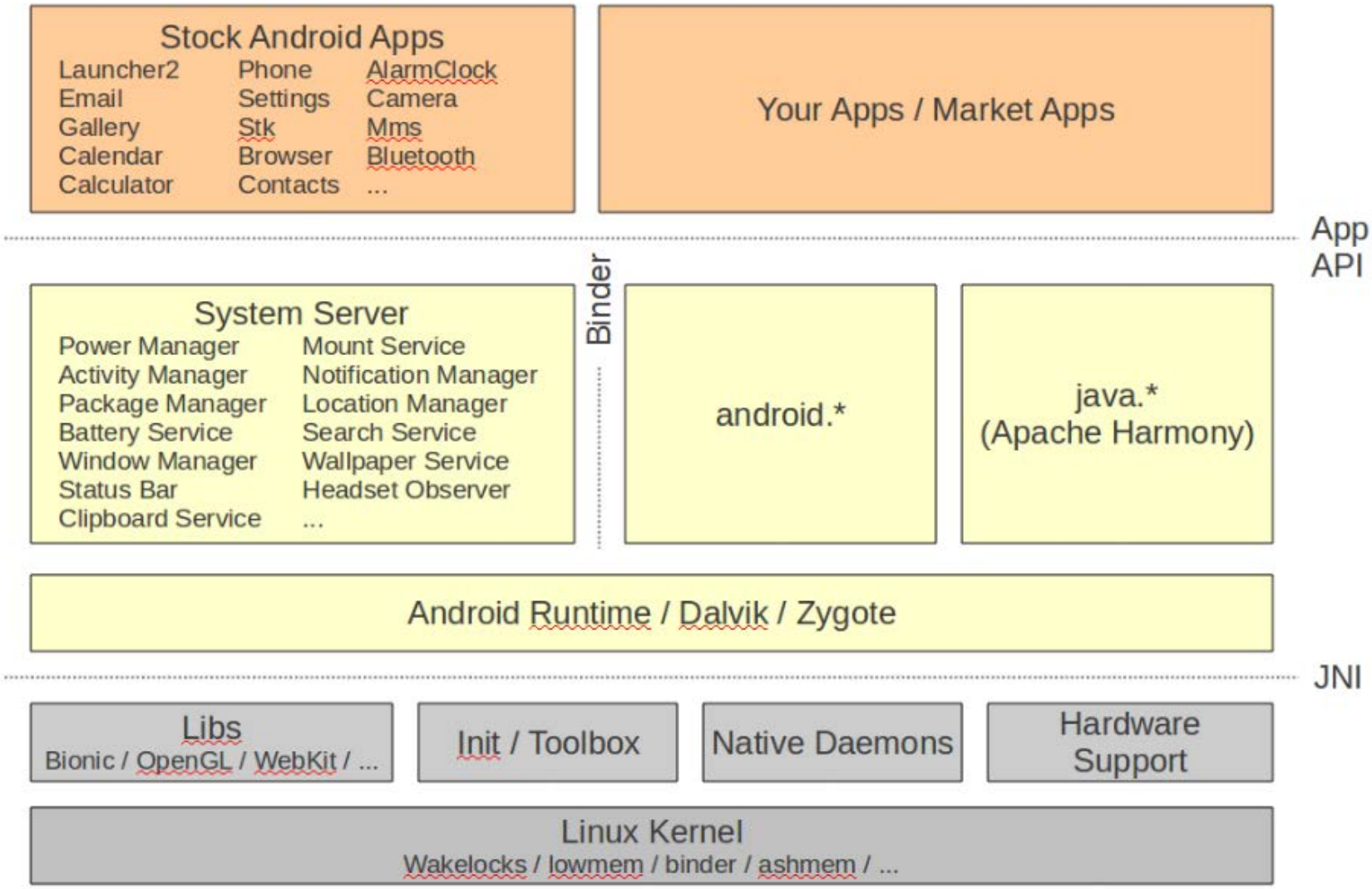
Mobile Market

- Lots of apps supported by advertisement ala shareware
- All the behind the scenes infrastructure, financial payments, statistics, marketing etc. is taken care of
- Developers typically get ~70% of the revenue after the app store takes their cut
- Apps are rated by their users
  - Users leave a rating and comments
  - Make sure your application is good!

Android Architecture

- Based on a small optimised Linux kernel compiled against Google's Bionic *lib*c library
  - Optimised for low clock speeds and small memory footprints
  - Bionic is non-standard so applications written for Linux cannot be ported to Android directly
  - The Bionic library is one of the Native interfaces to the Android architecture
- Android Kernel = Linux Kernel + extra power management functionality + Dalvik VM

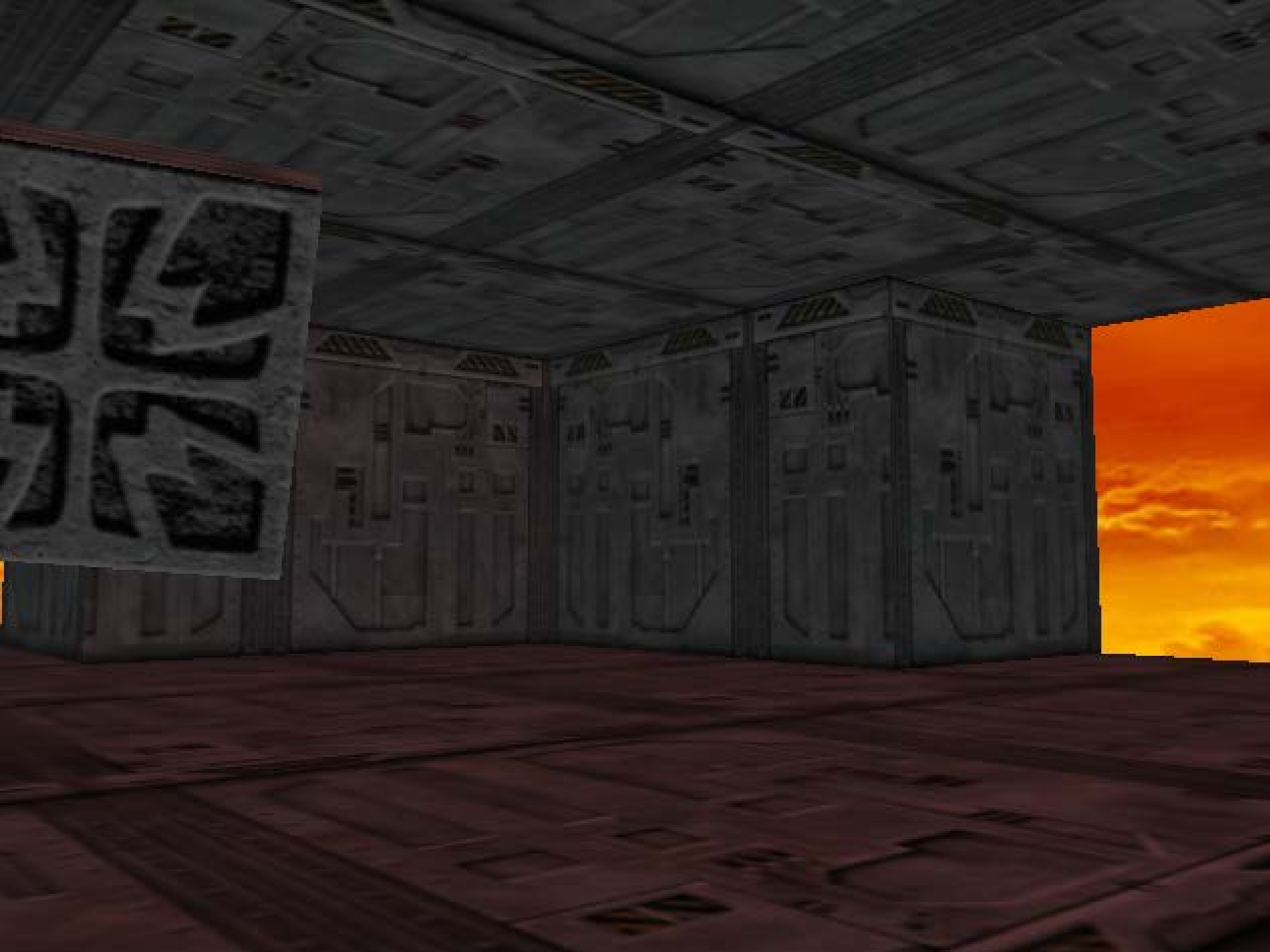
Android Architecture



Example Student Work

- …


SCORE: 0
Menu

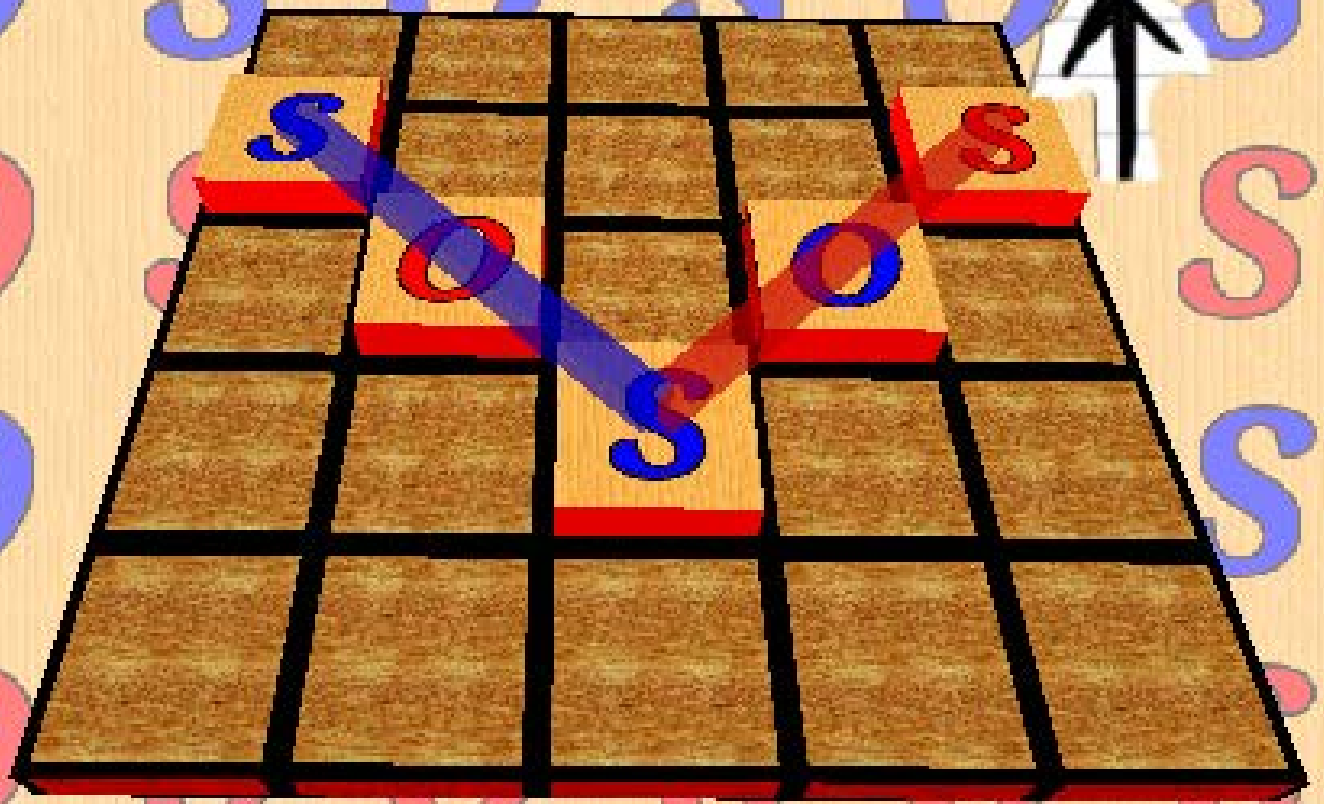


player

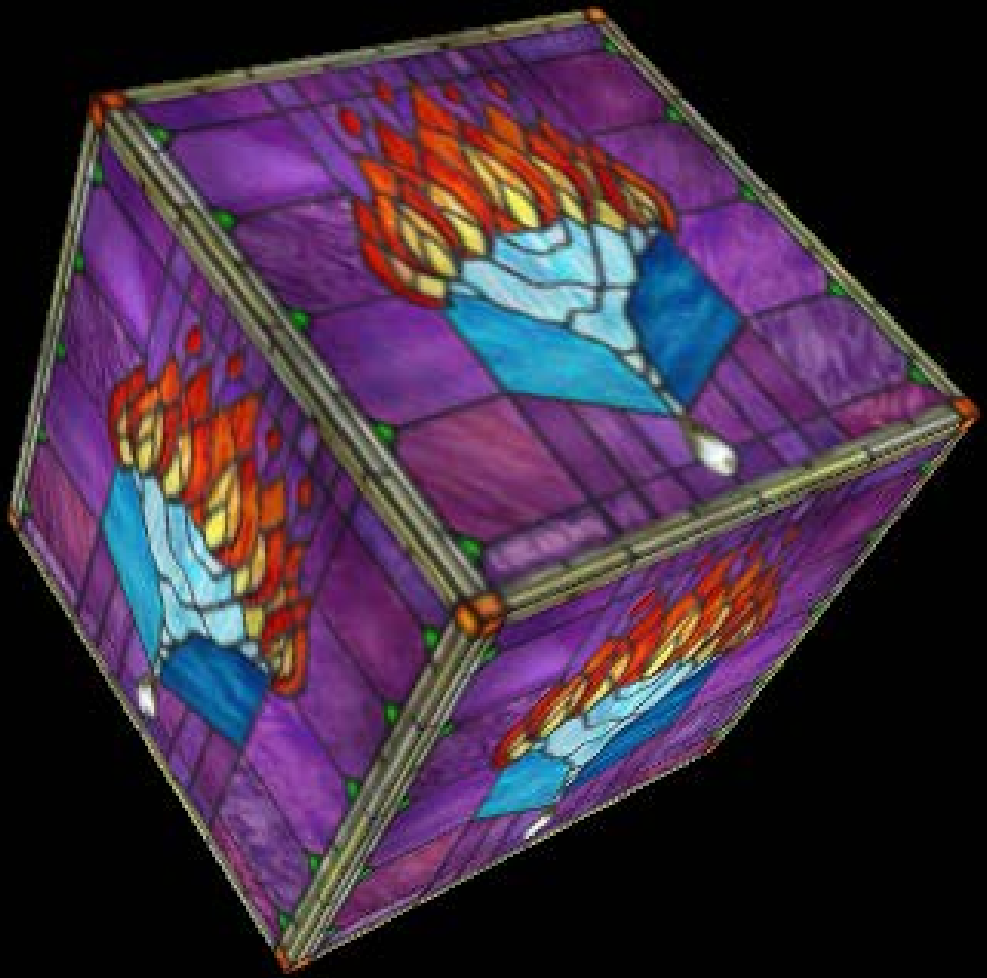
BLUE

player

RED



Adjust View



Class Room Help

- John Casey
- jcasey@unitec.ac.nz
- +64 9 815 4321 ext 6003

- Help and Advice with setting up AppInventor, HTML 5, Eclipse and your Android environment

- Questions?