Breakdown of the Course day

09:40 Arrive

10:00 Introductions

10:15 What is a Computer Program?

10:30 The simplest program

Ex 01

10:40 Top tips & Break!

10:45 Variables exercise

11:00 Simple program with variables

Ex 02

Ex 03

11:15 Functions exercise

Ex 04

11:40 Break!

11:50 Objects exercise

Ex 05

12:30 Top tips & Lunch

13:00 Visual Design Breakout session

13:45 Top tips & Break!

13:50 Recap

14:00 Build a 2D Shoot-em-up!

Part 1 - Background

Part 2 – Load Assets

Part 3 – Ship

15:30 Finish!

Teaching Programming Concepts

The following exercises teach basic programming concepts to 10-12 year olds. This is an experimental course that will be tailored once feedback is obtained. The concepts that form Part 1 of the course are as follows:

* Variables
* Methods
* Objects

Part 2 of the course follows the creation of a simple game, utilizing the concepts learnt from Part 1.

Repetition will be a core part of the 2 days. The above concepts must be ‘drilled’ into anyone learning computer programming. This will aid understanding of all subsequent lessons and will provide a solid beginning foundation for software development.

Breakout sessions – there will be 2 ‘breakout’ sessions to give the students a break from the ongoing topics and to look at an interesting and current area of software development. These sessions have been chosen to support the interest of the students and to give foundation to the course aspects. They are as follows:

* Hacking
* Design & Animation

# Variables

### Purpose:

To teach the concept of storage of values.

### Intro:

Any computer program needs to store values that it may need during its lifetime. For example, most computer games need to store the player’s score so that it can present it back to the player at any time. What are these values & how do we store them?

### Equipment:

Buckets, Card, Pens

### Method:

Hand out buckets to several of the participants.

Explain that each bucket is a program containing a single variable.

Each bucket should be labeled with what is effectively the name of the variable

Explain that each bucket will hold a certain piece or pieces of information

The variables are:

firstNumber = 1;

secondNumber = 2;

firstWord = "Hallo";

secondWord = "World!";

isWinter = false;

isHot = true;

theAlphabet = ['a','b','c','d','e'];

empty = undefined;

empty2 = null;

Ask each participant to find an appropriate value to store in their buckets. The values will be bundled together on a table or the floor. The participants will need to move, take a look and sort through the values, before returning with one for their bucket. They should then place the value in their bucket.

Go to Programming **Example 01 – Variables (http://codepen.io/peetj/pen/BoBpgv)**

# Functions

### Purpose:

To introduce the concept of an action performed by an object, ie. a method or function.

### Intro:

A method and/or a function are basically the same thing. Without methods, our programs would do nothing. For example, imagine a calculator program that couldn’t do any mathematical operations. (image of calculator with no buttons)

### Equipment:

Numbers on card, 2 people, ‘Calculator’ card

### Method:

Role play the Calculator by getting a volunteer to model the calculator. Tell the volunteer that they can’t actually perform the functions of a calculator. Then try and add 2 + 2. There should be no response.

Add a method ‘add’ to the calculator. This should be a role played by another volunteer. Try the sum again. This time the program should be able to call on the function – who has access to the numbered cards and can return the answer.

As an introduction to objects, explain that the calculator is a ‘thing’ or an ‘object’ that can do ‘stuff’ like add, subtract, multiply…etc

All objects will typically be able to do ‘stuff’ unless they are static ie. Just meant to be admired and don’t particularly do anything.

Go to Programming **Example 03 – Methods and Functions (http://codepen.io/peetj/pen/RWwxjJ)**

# Objects

### Purpose:

To teach the concept of objects as related to computer science.

### Intro:

Back in the day, objects would not have been one of the first things a budding computer programmer would have learned. However, this is now the 21st century so our approach will be slightly different to the traditional one!

### Equipment:

A variety of objects such as: apple, toy car, torch

### Method:

Ask the question ‘What is an object?’

Explain that an object has a structure and it may also do ‘stuff’

The structure of an object is generally called its **properties**.

The ‘stuff’ that it can do is generally called its **methods**.

Ask for 2 volunteers. One is blindfolded. The other has to describe an object to the blindfolded person who has to guess what it is.

Using the objects described above:

Ask the question, ‘Can anyone tell me the properties of…’

Ask the question, ‘Can anyone tell me the methods of…’

Let’s think about computer games:

Put up a slide of the classic computer game Space Invaders

Get the class to name some objects in the computer game

This leads us to our definition of objects:

“An OBJECT is a ‘thing’ that has properties and optionally methods”

# Object Definitions vs. actual Objects

### Purpose:

To illustrate that an object can be created from a template or definition.

Ask the question:

“Who knows the difference between an object definition/design and an actual object?”

At this point show some blueprint designs.

I could use some playdough and some cutters here!

Ask – “What is the only possible use of the (cookie) cutter?”

Answer: To create the shape/cookie.

The object definition in this case is the cookie cutter. The object or object(s) - because we can create as many as we want – is the cookie.

Go back to the Space Invader slide to illustrate this point.

Go to **Example 05 – What is an Object?**

# A crash course in HTML, CSS & Javascript

### What is HTML?

HTML is the structure of the web.

Every webpage in the world is built in HTML

Therefore if you know HTML you can build a web page!

### HTML Elements

HTML consists of elements. Everything in HTML is an element.

HTML elements look like this:

<p>This is a paragraph</p>

An element has an opening tag enclosed in less than & greater than signs.

It also has a closing tag which looks much the same except for the forward slash.

The element has some content which goes in the middle. In the above example, we are creating a paragraph element.

Browsers such as Chrome, FF, IE & Safari, know how to display HTML.

Let’s build a page!

# HTML

### Adding interaction with the outside world – methods/functions

Purpose: To explain methods/functions and how a program interacts with both itself and the outside world

Equipment: Paper/pen, Codepen

### Variables

Equipment needed:

Buckets

Numbers

Words

### Hello World

**Purpose**: *To teach the concept of variables*

**Scenario**: *The program should store a person’s name and say hallo to them.*

For this to work, we will need 1 bucket with a label ‘firstName’. One person is given the role of the program and one person the user. We also need 2 functions. One to store the name and one to repeat the name back to us. The steps are as follows:

1. User starts/invokes the program
2. User invokes the store function
3. The program runs the store function and stores the name away
4. User invokes the hello function
5. The program runs the hello function
6. The hello function invokes the getName function
7. The program says ‘hello <name>’ to the user

### Actors

User, main program, storeName function, getName function, sayHallo function, variable bucket

### Detailed Actions

The user prods the program who springs to life

The user then invokes the function on the program passing his name

The function stores the name in a bucket for later use

The user then invokes the sayHallo function on the program

The function invokes the getName function

The getName function retrieves the name from the bucket passing it to sayHello

The sayHello function then prints out ‘Hello <name>’