Deceptively Simple.
Insanely Fun.

MOBILE GAME DESIGN sounds like a pretty easy and sometimes scary word, but what does it mean to us as developers? Mobile games are a lot like old school game boy games. You have a small screen, and not exactly a super computer. Mobile games need to be

- Able to run on a small device
- Use the screen space properly
- Addictive
- Simple yet fun

Lots of games use the idea of **never reaching a game over.** The player wants to keep on keep on keep on playing. They do not have a set end to the game. We also want to include things like social interaction, and the ability to multi task and compete.
Below is a list of VERY popular games for mobile and computer. Can you tell me what they all have in common?

- Call of duty
- World of Warcraft
- Clash of Clans
- Hearthstone
- Minecraft pocket edition

- Geometry dash

Write the list below:

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.

We are going to now watch an awesome video about why games are addictive. Write blow some game design philosophies of the developers. https://www.youtube.com/watch?v=_BTGgCEFuQw.

Look at the picture on the front page of the hearthstone app on the phone. What does the app do well based on what we discussed?

Unity Game Engine is an open source AWESOME engine. You can make epic games for mobile apps and they are easily portable. We will be using c# to create some epic games. We will be reading and interpreting code to help you become an epic programmer. Below is some code we are going to talk about in a second. First let's talk about the basics.

Declaring Variables

```
        int i, j, k;
char c, ch;
float f, salary;
double d;
```

Initializing variables!

```
        int d = 3, f = 5;    /* initializing d and f. */
byte z = 22;         /* initializes z. */
double pi = 3.14159; /* declares an approximation of pi. */
char x = 'x';        /* the variable x has the value 'x'. */
```

Types of variables

```
using UnityEngine;
using System.Collections;

public class VariablesAndFunctions : MonoBehaviour
{
    int myInt = 5;


    void Start ()
    {
        myInt = MultiplyByTwo(myInt);
        Debug.Log (myInt);
    }



    int MultiplyByTwo (int number)
    {
        int ret;
        ret = number * 2;
        return ret;
    }
```

//end of code

Unity game engine

```
using UnityEngine;
using System.Collections;

public class IfStatements : MonoBehaviour
{
    float coffeeTemperature = 85.0f;
    float hotLimitTemperature = 70.0f;
    float coldLimitTemperature = 40.0f;



    void Update ()
    {
        if(Input.GetKeyDown(KeyCode.Space))
            TemperatureTest();

        coffeeTemperature -= Time.deltaTime * 5f;
    }
    void TemperatureTest ()
    {
        // If the coffee's temperature is greater than the hottest
drinking temperature...
        if(coffeeTemperature > hotLimitTemperature)
        {
            // ... do this.
            print("Coffee is too hot.");
        }
        // If it isn't, but the coffee temperature is less than the
coldest drinking temperature...
        else if(coffeeTemperature < coldLimitTemperature)
        {
            // ... do this.
            print("Coffee is too cold.");
        }
        // If it is neither of those then...
        else
        {
            // ... do this.
            print("Coffee is just right.");
        }
    }
}
```

```
     // start of onCollisionEnter

     using UnityEngine;
using System.Collections;
```

```
        public class ExampleClass : MonoBehaviour {
        AudioSource audio;

        void Start() {
              audio = GetComponent<AudioSource>();
        }

    void OnCollisionEnter(Collision collision) {
        foreach (ContactPoint contact in collision.contacts) {
            Debug.DrawRay(contact.point, contact.normal, Color.white);
        }
        if (collision.relativeVelocity.magnitude > 2)
            audio.Play();

    }
}
```

```
using UnityEngine;
using System.Collections;

public class MyGameClass : MonoBehaviour {
   void MyGameMethod() {
        // Message with a link to an object.
        Debug.Log ("Hello", gameObject);




       // Message using rich text.
        Debug.Log("<color=red>Fatal error:</color> AssetBundle not found");
    }
}
```

NOTES! Add any notes you want below