# Unity Basics

Unity is a game enige. We use Unity in combination with C#. We can highly recommend to use Visual studio (or maybe another modern IDE) over monodevelop that is packged with unity. There is some package integrating Visual Studio and Unity, install this package.

The general workings of Unity are well-explained on their own site.

I would recommend doing one or two tutorials. For example Roll-a-ball and 2DUFO.

https://unity3d.com/learn/tutorials

Make sure you understand at least the following:

* How to make a 2d game in unity (camera perspective, z-axis etc.)
* How prefab object work in unity
* How to changes scenes using a scripts

Furthermore their documentation is also very usfull:

<https://docs.unity3d.com/Manual/index.html>

# Unity tips and trick

This section contains general unity tips and tricks

* [Seralize field] makes a field inspectable from unity (even if it’s private)
* You can give the editor a color when it’s running. This is very useful because changes you make in the editor while the game is running are not preserved.
* Sometimes you have prefabs containing elements several levels deep (e.g text inside a button inside a canvas inside a game controller container object). Unity does not allow you to change prefabs this deep in the tree view and if you change some setting in a single scene and apply the change you might inadvertly make some locally changed setting global. The best solution is to make a special empty scene with a fresh prefab instance and to make your changes in this instance.

# Implementation Details

In this section we will give some specifics on how we implemented the game using Unity.

## Game/level design

For every gametype there is a prefab instance that is the controller for this game type. This controller object contains everything that is necessary to turn a empty scene (without a main camera) can be changed into a (empty/basic) level for this game type.

A nonexhaustive list of things that are provided by such a controller is given below:

* Background
* Main camera
* Game logic
  + E.g checking for victor/defeat and advancing the scene
* UI (User Interface)
  + E.g. enabling advance button when meeting victory condition

To expand such a empty/basic level one sometimes has to enter level points in the controller (artgallary) and sometimes has to add another prefab instance to the scene (Kings taxes, the divide)

## Advancing levels(scenes)

There are two different systems for advancing levels, depending on the (mini)game. Both methods are script based.

Either we attached a script to the button advancing a level. In this script we can set a variable indicating what the next scene will be.

Otherwise this functionality is provided directly in the controller. In this case the controller has a field in which you can fill in the name of the next scene.

# Adding levels

## Kings Taxes/Divide

1. Create a new scene in unity (and give it a logical name)
2. Add Scene to the build using “File>Build Settings…>Add Open Scenes
3. Delete the automatically generated Main camera
4. Add the prefab controller object
5. Add instances of the prefab game objects (soldiers/castles)
6. Change the nextlevel field of the previous level to this level and of this level to “agvictory” or change the nextlevel fields of the advance button if this game has a advance button.

## Art Gallery

1. Draw a Clockwise polygon using IPE
2. Create a new scene in unity (and give it a logical name)
3. Add Scene to the build using “File>Build Settings…>Add Open Scenes
4. Delete the automatically generated Main camera
5. Add the prefab controller object
6. Run the python sricpt addIpeLvl in the command line (details given in the script)
7. Change the numberOflighousesField to a suitable number
8. Change the nextlevel field of the previous level to this level and of this level to “agvictory”

Asses store packages we use

* Unity test tools

Other utilities

<https://www.reddit.com/r/Unity3D/comments/31enwc/free_playerprefs_utility_to_add_delete_modify_in/> (Alredy in the Assets/editor folder, includes readme)