**Topic: Unity3D Conceptual Framework**

**Overview:** Of course you want to dive right in, but experience suggests that taking time to read this very short document will cut down your search for information when you need it.

**What is Unity3D?**

Unity is referred to as a *game engine,* or a *3D engine.* In fact, it is both: you can create complex 3D with interactivity (e.g. a game) or without interactivity (e.g. an animated video). See <https://unity3d.com/unity>. Unity3D has three significant components:

1. An sophisticated ‘game’ engine that renders graphics, handles interaction (both user events and physics events). The engine operates behind the scenes; we don’t interact directly with it.
2. A GUI (graphical user interface) design environment that simplifies the more mundane aspects of project development, but allows flexibility through general purpose object-oriented programming in C#, JavaScript or Boo. The GUI is where all of the development takes place.
3. Online resources: manuals, tutorials, sample projects, and ‘live’ lessons (webinars). If you need to figure out something about Unity, the online resources will probably have the answer, though they may be difficult to navigate and decipher at times.

This quick start guide, intended for people working on the *Genderless In Germany* project, focuses on the tools and techniques necessary for creating a modified *visual novel* that is exclusively in 2D. Although Unity3D is *platform independent* we will initially focus on projects that produce games that can run in a browser.This will require rendering graphic images, establishing standard user interfaces (buttons, text-boxes, sliders), managing resource files, and interfacing with a tutoring system[[1]](#footnote-0). Consequently the following topics are minimally covered, or not covered at all, though you are encouraged to contribute by building guides for them:

* Component based animation.
* 3D design and efficient rendering
* Physics beyond simple linear motion (e.g. no rotation)
* Cross-platform deployment.

**An Object Framework**

A unity project is referred to as a *game,* if it includes user interaction, and is referred to as an *animated video* if it does not. Unity3D projects consist of a set of *objects* that *extend* the basic functionality of the *engine.* When you build a project you are defining *instances* of particular *object classes.* Examples of classes include *scenes,* and *game objects.* An instance of a class can have *components;*  both *data* (information about the object) and *methods* (‘actions’ that the object can perform) that elaborate on how it works.

In Object-Oriented Programming (OOP) code is written as a set of *classes.* A class organizes data and methods. An *instantiation* or *instance* of a class is an object that has specific values for that data, and can be invoked to execute one of its methods. Objects can be data for other objects in a *has-a* relationship. Classes can have subclasses, where the subclasses *inherit* or have access to the parent class data and methods. The *encapsulates* information so that all of the details are not overwhelming. Most coding in Unity3D is creating instances of classes. The GUI interface provides an efficient way to specify data, such as position, or velocity, as well as link objects together. The data associated with a game object is called a *component* of that object. Game construction involves creating game objects and giving specific values to the components. Sometimes that value will be a script (a method in OOP) to execute some task. This ability to move between data and methods (values and scripts) is why Unity3D is such a great design tool. However, it requires that you understand how to move gracefully between the GUI and coding. This set of resources is intended to provide you with the basics.

1. The pre-alpha system that will be built in Spring 2017 will imbed the tutoring system into Unity so we can get up to speed, but eventually the tutoring engine will be abstracted as an entirely separate. [↑](#footnote-ref-0)