Manual Unity Game Scripting

Term 1.4

Course Manual study year 2021/2022

Bachelor Creative Media and Game Technologies (CMGT)
School of Creative Technology

(Unity megacity screenshot – see https://unity.com/megacity)

Publication date	April 28th, 2022
Version	1.00
Module coordinator	Paul Bonsma (PBO06)
Lecturer	Rene Heijnen (HEJ), Yiwei Jiang (YJI01), Yvens R. Serpa (YRE03), Douwe van Twillert (TWI), Daniel Valente de Macedo (DVA06), Luuk Waarbroek (LWA08)
CMGT roles	Designer, Engineer

1 General overview

Module Name	Unity Game Scripting
Unit code	L.24225
Year and Term	1.4
CMGT roles	Designer, Engineer
Credits	3 ECTS
Lessons	6 x 2 hours lecture (12 hours), 6 x 3 hours lab (18 hours)
Study load	84 hours (=3 ECTS)
Responsible lecturer	Paul Bonsma (p.s.bonsma@saxion.nl)
Lesson structure	2 hours lecture, 3 hours lab each week
Module summary	In this module you will learn how to use the Unity engine and editor to create interactive prototypes, games and experiences. The focus lies mostly on creating functionality using scripting, enabling artists to insert their assets and polish scenes.
Industry relevance	Unity 3D has become one of the most widely used game engines, used to create games, interactive experiences, visualization tools and animations on all levels. The quality and amount of tooling available, and the high quality documentation, make it the tool of choice for many companies in the field of creative, interactive media.
Type of exam	Assessment, based on a product (a Unity project uploaded to Blackboard)
Exam code	T.50257
CMGT Competences	1. Technical research and analysis 2. Designing, prototyping and realizing 3. Testing and rolling out 6. Designing
Required prior knowledge and skills / conditions for enrolment	It is assumed that the student has basic programming skills, so he/she has successfully passed the course 1.1 Programming Basics.
Preparatory for:	This module prepares for many different courses in the second year. In particular, Unity will be the main tool for most of the second year projects.

2 Why this module?

Unity3D is a widely used tool to create interactive applications and games, in 3D and 2D. The quality and amount of tooling and free assets that are available for the engine is extraordinary, and continues to increase steadily, with regular updates. The rendering quality and efficiency of the engine can compete with any other game engine. In addition, it surpasses other game engines in terms of quality and amount of documentation and tutorials that are available for it. In all, this makes it the tool of choice for many companies in interactive media, and one of the main tools in the CMGT program.

Unity enables a good workflow for teams consisting of artists, designers, and engineers. As a designer, you can quickly prototype ideas, add game and interaction logic to scenes, and create interactive user interfaces. As an engineer, you can create tooling for designers and artists, and execute complex programming tasks (such as procedural generation). Both tasks require a good understanding of the Unity engine, editor, and component based programming using the Unity API.

The goal of this course is to give an introduction to Unity scripting, and Unity's main built-in tools. Of course it is impossible to get a complete overview of all of Unity's tools and tricks during one course, or probably even during a four year study(!), but this course will show what is needed to create a simple and complete game in Unity, using existing art assets. In addition, we will focus on writing reusable code and using prefabs and scenes correctly, to enable a fast and efficient team workflow.

2.1 What happens in the labs and lectures?

In each lecture, an aspect of the engine will be explained and demonstrated. This is mostly done by working in Unity on a simple demo.

During the labs, you can work on weekly assignments, related to the lecture demo, based on handout projects. During the later labs, you can also get feedback and tips on your own game, to be shown at the assessment.

2.2 How does this module relate to other modules in the CMGT study programme?

We expect that you have basic programming skills, so that you have successfully passed the 1.1 Programming Basics course (for designers or for engineers).

This module directly prepares for various second year courses, and most of the second year projects, which will use Unity.

3 What are you going to learn in this module (learning objectives)?

After completing this module successfully, you...:

- ...understand the Unity Editor and can use it to construct an interactive application.
- ...are able to use scenes, assets and prefabs.
- ...are able to develop basic game interaction using component based programming.
- ...are able to implement collision detection and construct physical interactions using the Unity Physics Engine.
- ...are able to design and implement a relevant resolution independent UI using the Unity User Interface.
- ...are able to polish the visual and auditory fidelity of an interactive application.

More detailed learning goals will be published weekly on Blackboard.

4 Which resources do you need?

The course will be taught using *Unity 2021.3.X* (currently, X=1f1). Make sure you have this version installed and working before the start of the course. *Tip: when installing Unity, make sure you include Visual Studio in the installation.* This will save a lot of hassle related to enabling code completion, which is essential for getting anything done!

These are the main sources of information on Unity:

- Manual:https://docs.unity3d.com/Manual/index.html
- Scripting API: https://docs.unity3d.com/ScriptReference/index.html
- Tutorials: https://learn.unity.com/

Good, specific tutorials:

- Scripting tips: https://learn.unity.com/project/beginner-gameplay-scripting
- A good introduction: roll a ball: https://learn.unity.com/project/roll-a-ball-tutorial or https://learn.unity.com/project/roll-a-ball
- Youtube: Brackeys channel: “How to make a video Game in Unity”. First video: https://youtu.be/IlKaB1etrik

As mentioned before, there are countless other good and free tutorials, for specific goals or on specific aspects of Unity, so feel free to search for additional tutorials that match your needs!

5 What does the programme of this module look like?

Every week there is a lecture, treating a different aspect of the engine. Details will be published weekly on Blackboard.

During the first weeks, lab assignments will be published on Blackboard, which help you to practice with the topics of that week. During the last labs, you can work on your own game (to be shown at the assessment), and receive tips and feedback for it.

6 How is this module assessed?

Since this is a 3ects course (84 hours), it is expected that you work outside of the lectures and labs, e.g. by finishing the lab assignments, following extra tutorials, experimenting with the engine, reading the official documentation, etc.

6.1 Assessment and redo

There will be an assessment in week 4.9, and a redo assessment in week 4.11, where you present a self-made 3D game in Unity. In order to participate, it is necessary to upload this Unity project on Blackboard, at least a day before the first assessment. During the assessment, you should show your game, explain your project/scene/code setup, and answer questions related to the learning goals.

During the later labs, you can already show your project, and possibly “sign off” parts of the rubrics, since time is limited during the assessment itself.

6.2 Grading

The rubrics are included at the end of this manual. If the preconditions are not satisfied, the grade will be a 1. Otherwise, it’s the total number of points, divided by 10.

7 Who are the contact persons for this module?

Module coordinator / lecturer:
Paul Bonsma (PBO06) p.s.bonsma@saxion.nl

Additional lecturers:
Rene Heijnen (HEJ)
Yiwei Jiang (YJI01)
Yvens R. Serpa (YRE03)
Douwe van Twillert (TWI)
Daniel Valente de Macedo (DVA06)
Luuk Waarbroek (LWA08)

8 Rubrics

Rubric Unity Game Scripting

Preconditions:

- On Blackboard, a **3D game** made in Unity is handed in; both the Unity project (mandatory) and a PC/Mac build (optional).
- The Unity project upload is at **most 250MB, and does not contain any temporary files or folders**.
- All art assets (audio, models, textures, sprites, animations) used in the project are free or self-made.
- All C# scripts are self-made (possibly with help of a lab teacher), or from the Blackboard handouts. (This means: You may use e.g. online tutorials for inspiration, and get feedback and tips from fellow students, but don't copy/paste significant pieces of code from external sources. In other words, you should be able to rewrite your code from memory. To keep the grading clear and fair, don’t include tooling assets (the ones that include code) from external sources such as the asset store.)

If you don't meet the preconditions the assessment will be assessed with a 1.

Otherwise, your grade is the number of points divided by 10.

Rubrics explanation: most of the rubrics below are formulated as “the student shows...”. This means: the feature should be part of your project, *and* you should be able to proactively explain the feature and answer questions about your setup!

	Insufficient	**Sufficient**	**Good**	**Excellent**
Editor & Creating interactive application (15%)	**3 pts** The game contains serious bugs, is unplayable, or unclear without extra explanation, or there is no working build uploaded.	**9 pts** The student creates a simple, working game build that is playable without the student's assistance.	**12 pts** S + at least one of the excellent criteria satisfied.	**15 pts** S + The student creates a game with (1) interactive help / tutorial, (2) elaborate scenes and/or good level design, (3) interesting game progress (e.g. multiple levels, difficulty system, inventory)
Scenes (10%)	**2 pts** No scene changes, insufficient interaction between different game objects, or insufficient understanding.	**6 pts** The student shows how game objects are linked when loading the scene (e.g. enemies chasing a player), and how scenes are changed (e.g. from start screen to game level to end screen).	**8 pts** The student shows multiple methods for linking game objects at scene load, and how information can be maintained during scene transitions (e.g. player score, sound settings).	**10 pts** G + The student clearly explains the (dis)advantages of these different methods.
Prefabs (10%)	**2 pts** The project does not contain multiple instances of self-made prefabs, or the prefab workflow is not sufficiently explained.	**6 pts** The student shows how self-made prefabs are used to improve the workflow.	**8 pts** S + The student shows how overrides are effectively used, and prefabs are created and destroyed at run time.	**10 pts** G + The student shows how prefabs can be used extensively to improve the workflow for editing multiple (similar) levels. Prefab variants are used.

Component based programming (20%)	**4 pts** At least one of the sufficient criteria is not satisfied.	**12 pts** The student understands component based programming, knows the basic Unity API commands and event functions (Awake, Update, etc), and writes reasonably clear code without excessive repetition.	**16 pts** S + half of the excellent criteria satisfied.	**20 pts** S + The student shows how to (1) write robust code (including null checks), (2) write reusable code (e.g. using inspector properties, atomic components, inheritance), (3) write clear code (code conventions, comments, naming), and (4) the student has a good overview of the core Unity API.
Physics (10%)	**2 pts** Insufficient or wrong use of basic physics elements	**6 pts** The student shows how the physics engine is used to correctly move objects and create interactions between game objects. Basic physics elements are used (rigidbodies, colliders).	**8 pts** The student shows multiple different, effective ways of moving objects and creating interactions using the physics engine. Intermediate physics elements are used (e.g. raycasts, physics materials).	**10 pts** G + The student explains the best way for moving objects and creating interactions in different cases. Advanced physics elements are used (e.g. compound colliders, physics constraints, collision layers).
Transforms (10%)	**2 pts** No good demonstration or understanding of rotation or parent/child relations	**6 pts** The student shows how to rotate objects by code, and explains parent/child relations.	**8 pts** The student shows how to manipulate Euler angles and use rotation pivots.	**10 pts** The student shows how vectors, quaternions and parent/child relations can be used to create custom behavior (e.g. animation by code, complex player controls).
UI (10%)	**2 pts** There is insufficient UI, too few different UI elements, or insufficient understanding of the UI system.	**6 pts** The student shows how basic UI elements (Button, Text, Image) are effectively used to display game data and allow user input.	**8 pts** S + The student shows how the UI elements are grouped and placed to create a UI that works well for different aspect ratios, and explains RectTransforms.	**10 pts** G + effective use of multiple advanced UI features (e.g. dragging, advanced UI elements such as text inputs, scrolling, button selection with keyboard, world space UI, etc).
Polish (15%)	**3 pts** Insufficient audio or lighting, or insufficient understanding of these aspects.	**9 pts** The student shows how audio feedback is used to improve user feedback, and how basic lighting is used to increase visual fidelity (at least two light types).	**12 pts** S + at least two of the excellent criteria are satisfied.	**15 pts** S + The student shows how the scene and user experience are enhanced using (1) particle effects, (2) audio effects created with a mixer and/or reverb zones, (3) (UI / character) animations, (4) advanced lighting or post processing.