Motion Engine: Creating Wii-like Games for PC

Samuel Truman[1]

***Abstract*—Starting a few years ago, motion input devices which allow the recognition of real-life, 3D gestures, began gaining popularity and are offering unique gameplay opportunities. While present on every modern living room console, their use with PC has been confined to Virtual Reality. Motion Engine is a game engine which allows fast prototyping of 3D video games using the Nintendo Wii Remote as motion input device. An easy-to-use Entity-Component-System saves a lot of time, since developers don't have to worry about complex low-level matters like rendering, physics, memory, or asset management. Motion Engine is well-documented, tested, and can be easily extended. This paper discusses the benefits of Motion Engine and its contribution to PC game development.**

I. Introduction

In recent years, an exciting new trend can be seen in the games industry. Traditionally, video games make use of input methods such as mouse and keyboard, joysticks, or gamepads. However, new means of input have joined those traditional input devices. Innovations and novel means of play have been thriving through the use of 3D motion input devices. This development, however, has been mostly limited to consoles and big-budget game studios with the resources to develop for consoles such as the Wii. Console vendors often choose not to publish their game engine and development tools to the public but set up programs requiring developers to sign-up and be approved in order to gain access to SDKs and other development resources. This presents a significant obstacle to indie developers and educational institutions to develop for those platforms.

Motion Engine is a 3D game engine that allows rapid prototyping of games which utilize the Nintendo Wii Remote (Wiimote) as a 3D input device for PC (see fig. 1). Motion Engine is well-documented, tested, and easily extensible. A built-in Entity-Component-System fosters clean and low-coupled code in both engine and games. The engine is accompanied by two demo games which demonstrate possible usages of the features provided by the game engine and the Wiimote. Dogfight (fig. 4) is a family-friendly fighter plane game in which the player controls the plane by tilting the Wiimote. The goal is to shoot as many balloons as possible. Dogfight employs almost exclusively the Wiimote's acceleration sensor as input. In Target Training (fig. 3), the player's reactions are put on trial by challenging him or her to shoot targets as soon as they appear. The player aims by pointing at the PC screen utilizing the Wiimote's optical sensor and the Wii Sensor Bar. This demo is especially interesting to demonstrate the Wiimote's pointing capabilities, which are not supported by most other motion input devices.

The Wiimote was chosen as the main input device due to several properties appealing to both developers and consumers. The Wiimote costs only about €20 (including accessories) [1] rendering it significantly cheaper than alternative motion input devices. Still, the Wiimote's infrared (IR) camera tracker offers high resolution at a 100 Hz refresh rate and integrated hardware object tracking [2]. The Wiimote exceeded the Wii's lifetime, being supported by its successor, the Wii U, and heavily influenced the JoyCon, which is the controller for Nintendo's contemporary console, the Switch. However, the Wiimote seems to still remain popular, as Just Dance 2020 was just announced for the Wii [1].

Fig. 1: Wii Remote

Fig. 2: Nunchuk

To provide a foundation to this discussion, relevant related work will be addressed in the next section, starting with the introduction and comparison of different motion input devices. Afterwards, Motion Engine is discussed in greater detail in section III. In section IV, results, benefits, and challenges are discussed. Finally, a brief outlook on potential future work is provided in section V.

II. Related Work

In 2006, Nintendo released their very successful console, the Nintendo Wii. The Wii's unique characteristic was the focus on a newly introduced input device, the Wiimote. The Wiimote most notably allows 3D motion recognition by means of an acceleration sensor and pointing by means of an optical sensor. A newer version also includes a gyroscope. In the following years, similar motion input devices were developed for every major console but the PC. In 2010, Sony released the PlayStation Move controller, which allows motion sensing and position tracking by means of an optical sensor called PlayStation Camera and a magnetometer. PlayStation Move can also be used for PlayStation VR (PS VR) and starts at about €70 [3]. Only one month later Microsoft released the Kinect for their Xbox console, which allows controller-less motion sensing by utilizing optical sensors. Microsoft has stopped manufacturing the Kinect meanwhile [4]. For PC, no standard motion controller is available. Motion controllers are only slowly gaining popularity with Virtual Reality (VR) becoming mainstream. The two by far most popular VR systems for PC, the HTC Vive and Oculus Rift, both come with their own set of motion controllers. These controllers,

[1] `samuel.truman@stud-mail.uni-wuerzburg.de`

however, are - like VR hardware in general - quite expensive and generally not utilized in non-VR games.

While there are obviously plenty of games for the Wii extensively utilizing the Wiimote, non-Wii games utilizing the Wiimote are effectively non-existent. The long list of Wii games covers all genres. Sports games seem to be especially popular. Most notably Wii Sports [2], which is considered one of the most successful video games in history with more than 80 million copies sold [5]. Although many Wii games are highly optimized, they are limited by the hardware capabilities of the 2006 released Wii.

The most popular modern-day game engines generally don't support the Wiimote as input device out of the box. A Wiimote plugin for Unreal Engine seems to have existed at some point, however, it doesn't appear to work on contemporary Unreal versions and was removed from the plugin explorer. For Unity, there are two plugins: Unity-Wiimote [6] and WiiBuddy [7]. The former seems to have some technical difficulties which could not be fixed for years, esp. regarding the Nunchuk extension. The latter has generally good reviews and costs only €22.33. Some reviews, however, suggest usability issues.

III. METHODOLOGY

In this section, Motion Engine is discussed in greater detail including development, use-cases, and benefits. The development was generally driven by the principles "performance over memory" and "usability over performance". This means, speed-memory trade-offs were mostly decided in favor of speed at costs of higher memory usage, and simple, maintainable code was favored over cryptic yet fast code. Motion Engine can be used to build games for Windows using OpenGL 4.2 and modern C++17. The target compiler is MSVC 15 (2017) x64, but MSVC 16 (2019) and gcc will work, too. MinGW is currently not supported anymore due to a file size limitation causing problems with a third-party library in debug mode. Supporting additional target platforms in the future should be possible without much effort. Motion Engine can already be compiled and used to run unit tests under Linux, e.g. in a Continuous Integration (CI) pipeline. Such a CI pipeline was used extensively throughout Motion Engine's development and unit tests were added to ensure functionalities on different compilers and permit regression testing. During configuration in CMake, unit tests and a test project can be enabled or disabled.

A. Wiimote

Both the example games included (Dogfight and Target Training) highlight the use of the Wiimote, Motion Engine's primary input device. The classic Wiimote and refined version are both supported. Although not used by the example games, the Nunchuk (fig. 2) and Motion Plus extension - which adds a gyroscope - are equally supported. Motion Engine automatically disables Wiimote functionalities which are not in use in order to minimize data transfer and increase battery lifetime. For example, the IR sensor is disabled until pointing input is required. In order to use the Wiimote as a pointing device, a Wii Sensor Bar is required. Either the official one can be used or a non-official USB variant which costs about €8 can be used. There are also slightly more expensive wireless sensor bars. To connect to the Wiimote, the PC running the game is required to have Bluetooth. For PCs without Bluetooth, a USB Bluetooth dongle can be used. Bluetooth dongles start at about €5. Apart from the Wiimote, Motion Engine provides full support for mouse, keyboard, and contemporary Xbox and PlayStation controllers. Another important engine feature is the Entity-Component-System which is introduced next.

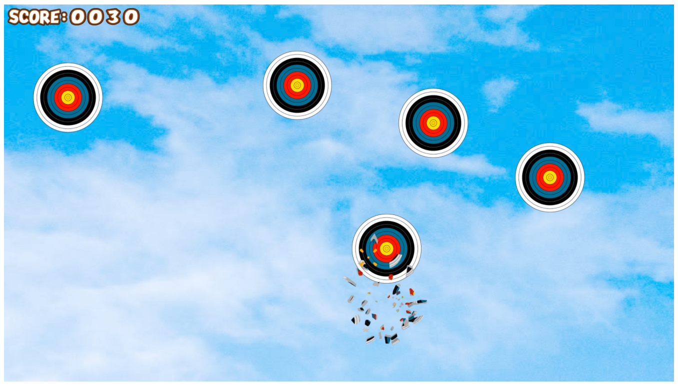

Fig. 3: Point with the Wiimote to shoot targets in Target Training

B. Entity-Component-System

The Entity-Component-System (ECS) is used throughout the engine and has a significant impact on engine and game code alike. The basic idea behind the ECS is to favor composition over inheritance and separate data from logic. Entities are essentially just an ID representing an object in the game world. A component is a set of data describing a specific aspect of the entity. For example, a Transform component stores data regarding the entities position, orientation, and scale in the game world. Systems use the data specified by components to achieve a specific task - in general periodically. For example, the physics system uses all physics data to update the physics simulation. Most major sub-engines like Rendering and Audio are systems. This approach is highly extensible, as new components and systems can be easily added without leading to deep inheritance hierarchies or high-coupled code. Also, this approach is highly adaptable, as switching e.g. to a different renderer would only require replacing the concerned system, without the need to change any other engine or game code. More on how the ECS can be utilized to quickly create games is explained in section III-D. A detailed explanation of how to use the ECS, as well as its implementation, can be found in the manual and API documentation.

C. Additional Features

This section briefly highlights some of the features which are slightly less obvious and might not be recognized just by looking at the example games. Nevertheless, they contribute to the workflow and game experience. A scene consists of a set of entities with their attached components, as well as a few special objects such as the main camera or skybox. A scene with its contents is serializable and can be saved to and loaded from JSON. Alternatively, a binary format based on CBOR can be used, which yields significantly smaller file sizes. Motion Engine also offers a set of post-processing effects which can be used to visually enhance games. A Sequence of several post-processing effects are for example used in Dogfight, the moment the plane is spawned. Motion Engine's standard rendering system uses Blinn-Phong shading and supports point-, spot-, and directional lights. The specular reflection of the sun can be seen e.g. on the plane in Dogfight, depending on the plane's position relative to the sun. Rendering techniques such as blending, normal mapping, anti-aliasing (MSAA), and gamma correction are supported and used in both example games. Every major 3D model format can be imported in Motion Engine, including FBX, OBJ, DAE, and blend. Most of the important image formats, including PSD, and several audio formats can be used. To draw UI, sprite and text rendering, along with dear imgui widgets can be used.

Fig. 4: Control a fighter plane with the Wiimote in Dogfight

D. *Workflow*

To create games using Motion Engine, a new CMake project must be created. Motion Engine needs to be downloaded and linked. Then, a game class can be inherited from the engine class and executed in the main function. The class interface provides callbacks which can be used e.g. to run code relevant to all scenes. Scenes can be registered to the scene manager or loaded from files. A scene can be filled with entities using the entity manager. Each entity represents an object in the game world. To change an entity's behavior, components can be added to the entity. Motion Engines provides several components, e.g. different lights, colliders, UI elements, and of course transforms, rigid bodies, and models. Furthermore, one can add own components or logic to entities. These components need to implement the Behavior interface. Adding behaviors to entities via lambdas is also possible. This is especially useful for quick prototyping. More on how to get started with Motion Engine including example code can be found in the quick start guide and the manual.

IV. CONCLUSION

Motion Engine is a well-documented, easy-to-use game engine which allows fast prototyping of 3D video games. Special thought was given to support creating applications which make use of the Wiimote and Nunchuk as an input device for gestures and pointing. Developers using Motion Engine don't have to worry about writing OpenGL rendering code or other low-level systems like memory or asset management. Engineers can conveniently extend the engine by adding new components and systems to achieve any conceivable task. The manual contains sections dedicated to engineers explaining how. Non-expert C++ developers should be able to build a simple prototype in just a few hours. Currently, Motion Engine has a 42.3% test coverage. This, however, includes OpenGL rendering code, which is generally difficult to unit test. Excluding this code in coverage reports would yield a significantly higher coverage but defeat the purpose of coverage reports. With VSync disabled, Dogfight has an average of about 120 frames per second (FPS) and Target Training about 150 FPS. With VSync enabled, both games have a stable framerate of about 60 FPS. These tests were conducted on contemporary gaming hardware (Nvidia GTX 1080). Utilizing a few rendering tricks, the framerate could be improved significantly in the future.

V. FUTURE WORK

This section briefly mentions how Motion Engine could be improved in the future. Several rendering techniques were initially implemented, but as they had no use for the example games, support was dropped. Re-adding them, however, could be done with minimal effort. These techniques include SSAO, instanced rendering, parallax mapping, HDR, and (omnidirectional) shadow mapping. An editor could be added to speed up the engine workflow as well as make the engine more accessible to designers. Based on the UI sprite rendering system, a dedicated 2D mode could be provided. The main yet missing requirement for such a 2D mode is probably a physics system focused on 2D games. The capabilities of Nintendo's latest motion controller, the JoyCon, could also be explored in the future. Additionally, machine learning algorithms could be utilized to foster smarter gesture recognition. Finally, a binary asset format could be introduced, as required by many providers of free assets on the Internet.

LUDOGRAPHY

[1] Ubisoft, "Just dance 2020," 2019.
[2] Nintendo, "Wii sports," 2006.

REFERENCES

[1] Amazon, "Wii Motion Plus Controller," https://amazon.de/dp/B07CPQGRB8/.
[2] J. C. Lee, "Hacking the Nintendo Wii Remote," *IEEE Pervasive Computing*, vol. 7, no. 3, pp. 39–45, Jul. 2008.
[3] Amazon, "PlayStation Move Motion-Controller," https://www.amazon.de/PlayStation-Move-Motion-Controller-Single-Pack/dp/B004V7K0IK.
[4] M. Wilson and M. Wilson, "Exclusive: Microsoft Has Stopped Manufacturing The Kinect," https://www.fastcompany.com/90147868/exclusive-microsoft-has-stopped-manufacturing-the-kinect, Oct. 2017.
[5] "IR Information : Financial Data - Top Selling Title Sales Units - Wii Software," http://www.nintendo.co.jp/ir/en/finance/software/wiiu.html.
[6] A. Biagioli, "An easy to use interface between Unity3D / C# and a Wii Remote controller.: Flafla2/Unity-Wiimote," Jul. 2019.
[7] "WiiBuddy - Asset Store," https://assetstore.unity.com/packages/tools/input-management/wiibuddy-4929.