GNG2101 Report

Project Deliverable – J

User Manual

Submitted by Group Z5:

David Neptune, 8272353

Philipe Rosier, 30005564
George Roumelis, 300115535
Briar Vandenberg, 300058322

July 19, 2020

University of Ottawa

Abstract

This paper outlines the all the important details on the creation of the milling simulator. In the first sections of the user manual, the different modules of the product are described. Subsequent sections discuss the creation of the product and the necessary steps needed for patch updates. Details about the different game objects and scripts are fully explored in this guide. This document further informs about any healthy and safety precautions, important troubleshooting issues and future update recommendation for anyone who would like to take the mantle on this simulator.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
GO	GameObject. The base class for all entities in Unity Scenes.
DRO	Digital Readout

1 Introduction

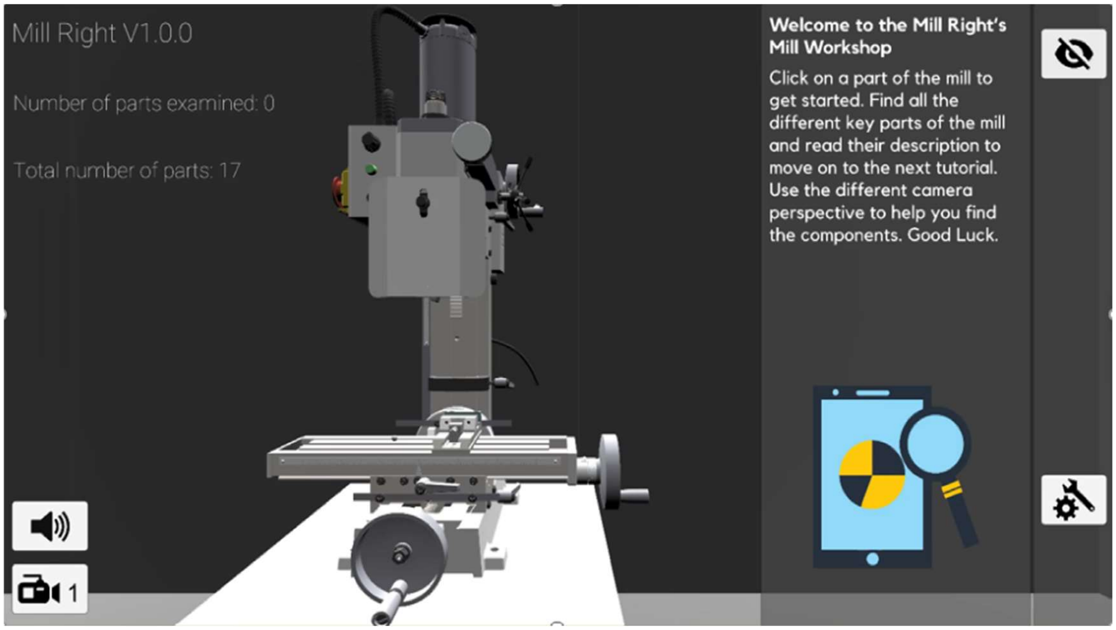
The content of this report documents the information about the final prototype. It encapsulates the purpose and function of each aspect of the product. The individual aspect of the product is explained thoroughly to guide users in the process and steps taken to build the simulator. The guide explains the purpose and format of each tutorial module, it also further details each important subsystem and their objectives. Furthermore, the user manual also desalinates the installation, maintenance, and operation necessary for the use of the simulator. The safety and health precautions as well as the troubleshooting are also duly outlined in this guide. Finally, the user manual has a section present which discusses future considerations for the product. The goal of this document is to inform and detail all the components to the production and continuation of this simulator.

2 Product Features

2.1 Tutorial Modules

2.1.1 Part Identification Module

This module encapsulates all the different key parts of the milling machine. The user can interact with the mill directly. When a part of the mill is interactable, it will change color on hover. When the user clicks on that part, a short description of the component and its purpose will appear on the panel (on the right-hand side). The module is setup so that user must click on each interactable part of the mill to move on to the next tutorial. This setup was chosen to force the user to read the description of each key part of the mill.


Mill Right V1.0.0

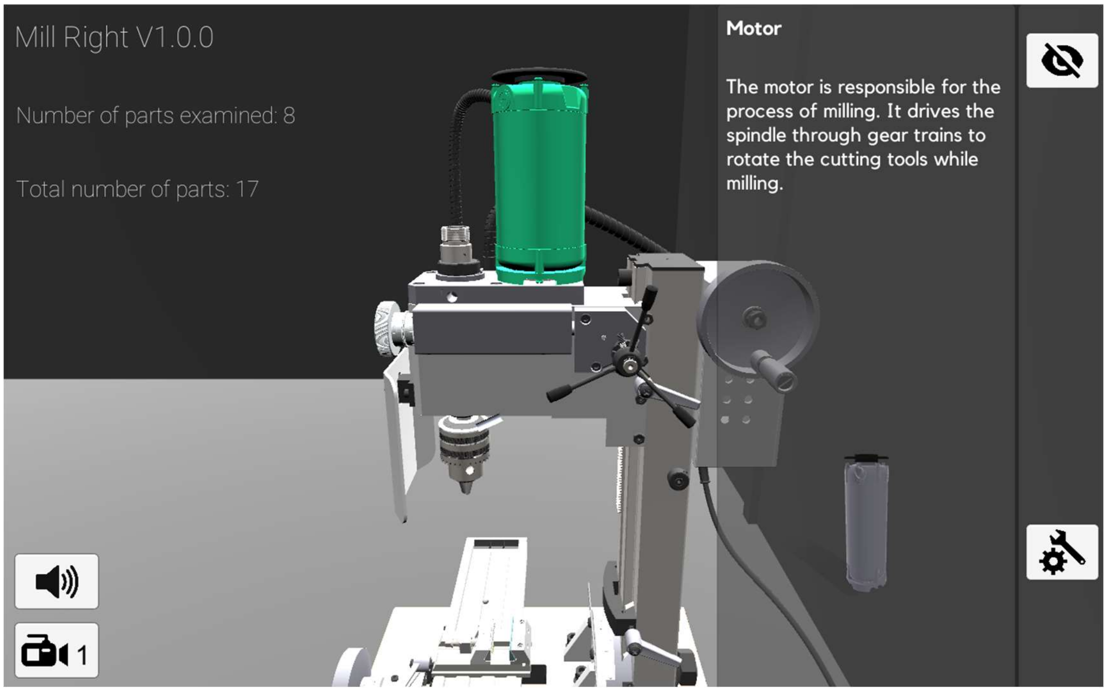
Number of parts examined: 0

Total number of parts: 17

Welcome to the Mill Right's Mill Workshop

Click on a part of the mill to get started. Find all the different key parts of the mill and read their description to move on to the next tutorial. Use the different camera perspective to help you find the components. Good Luck.

Figure 2.1 Mill Scene (Part Identification Module) display of initial scene setup


Mill Right V1.0.0

Number of parts examined: 8

Total number of parts: 17

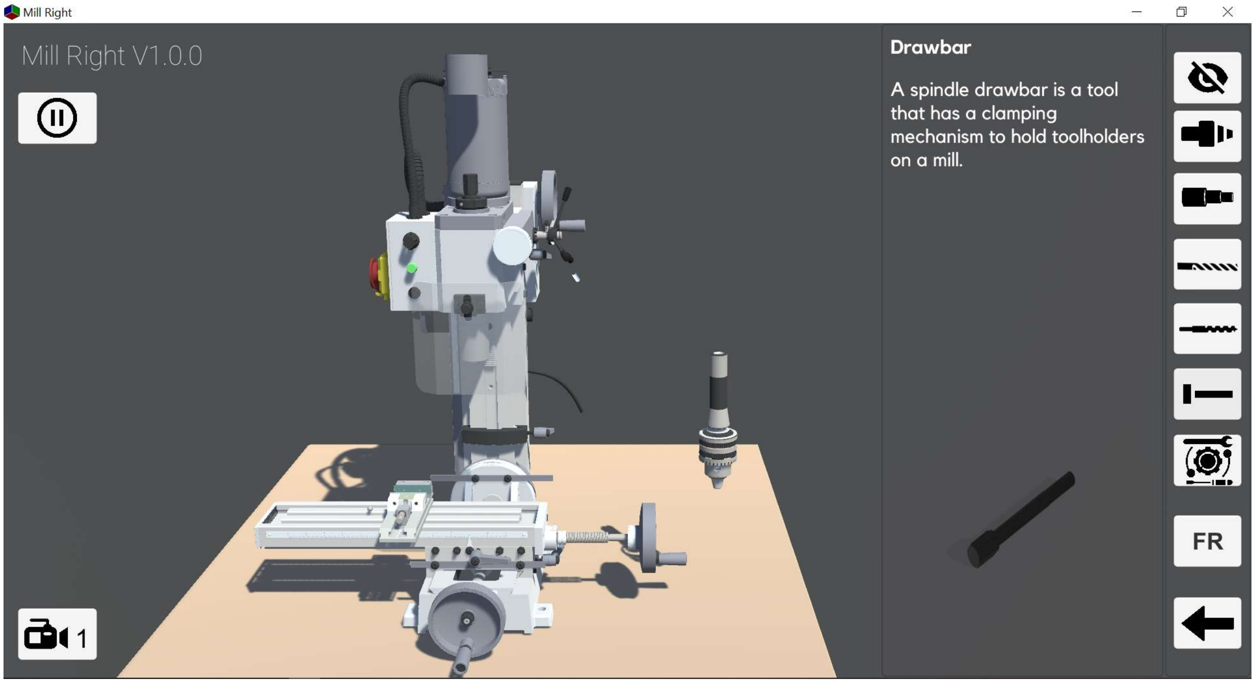
Motor

The motor is responsible for the process of milling. It drives the spindle through gear trains to rotate the cutting tools while milling.

Figure 2.2 Mill Scene (Part Identification Module) display of scene after clicking on mill motor (shows motor description)

2.1.2 Tool Holding Setup Module

The tool holding setup module focuses on the how-to setup a tool holder in a mill. An animation shows all the different steps necessary to interchange those tools. The module is accompanied by details of the different parts that are present in the tool holding setup (spindle, drawbar, etc.). This module is setup so that user can click between the description of the different tools. Furthermore, it also features a pause/play button for the user to control the animation.



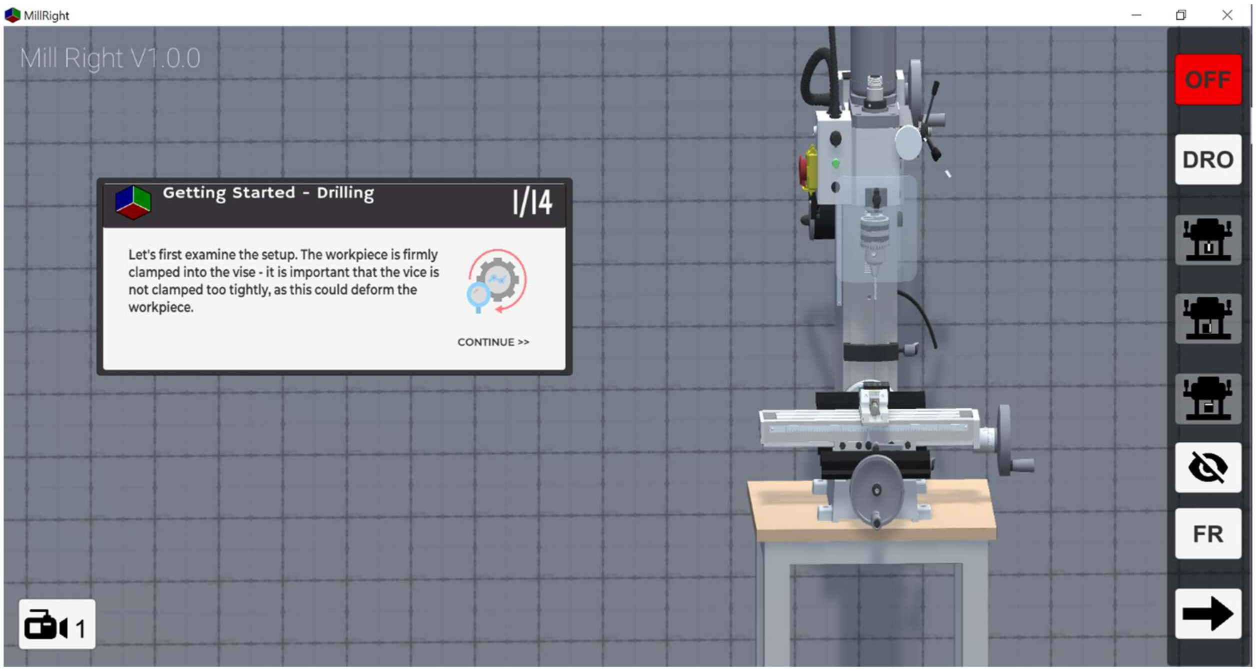
Drawbar

A spindle drawbar is a tool that has a clamping mechanism to hold toolholders on a mill.

Figure 2.3 Spindle Scene (Tool Holder Setup Module) display of scene during animation and after clicking on drawbar (shows drawbar description)

2.1.3 Milling Operations Module

This module's goal is for the user to learn how to setup and perform the different operation possible on milling machine. The operations featured in this module include drilling, side milling and face milling. The user can control the mill to perform these operations using the digital readout panel (DRO Panel). The dialogue boxes are present to help guide the user to perform the operation efficiently. To control the mill, the user has access to a digital readout panel which lets the user control the different axes, quill feed and the fine adjustments. More details on the DRO panel will be made available in further sections.



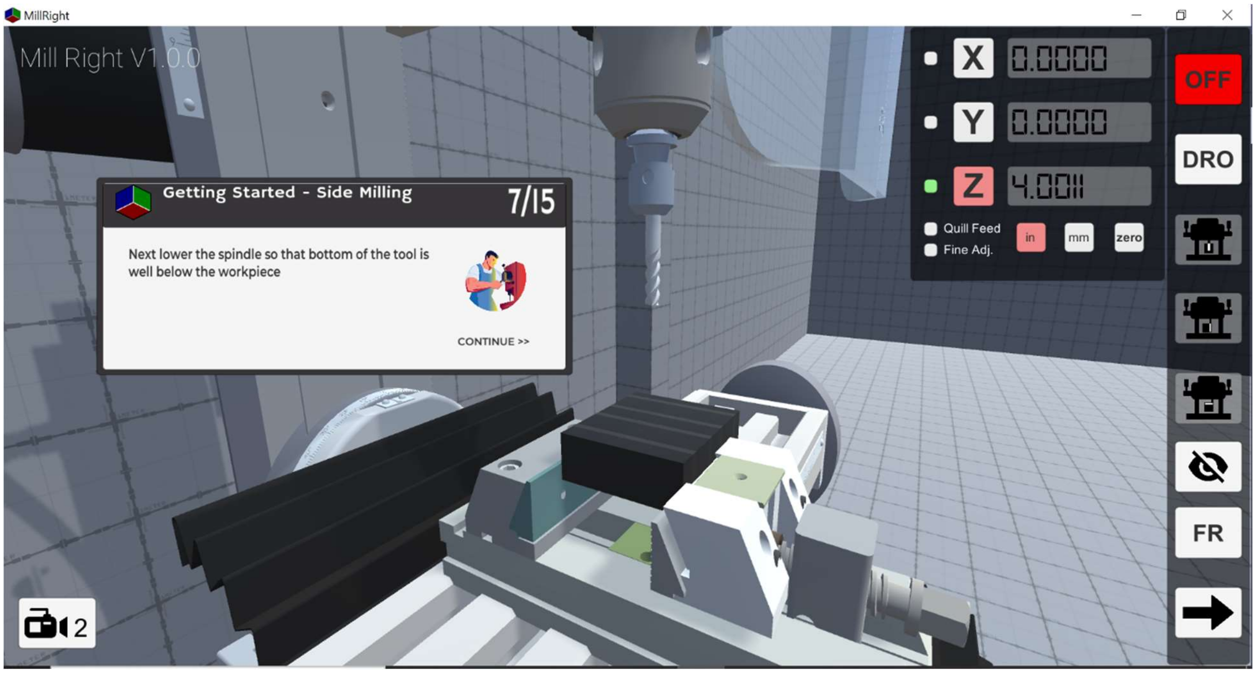
Getting Started - Drilling

1/14

Let's first examine the setup. The workpiece is firmly clamped into the vise - it is important that the vice is not clamped too tightly, as this could deform the workpiece.

CONTINUE >>

Figure 2.4 Intro Tutorial Scene (Milling Operations Module) display of scene during start of drilling guided lesson



Getting Started - Side Milling

7/15

Next lower the spindle so that bottom of the tool is well below the workpiece

CONTINUE >>

Figure 2.5 Intro Tutorial Scene (Milling Operations Module) display of scene side milling during guided lesson

2.1.4 Milling Simulator Module

The simulator module of the tutorial is the segment that allows the most control to the user. The user can change between different end mills, drill bits (which also implies between end mill holder and drill chuck). The user just like the operations module has full control of the DRO Panel. The user also has access to a schematic panel with the plans for the workpiece the user will mill. The purpose of this module is to allow the user the chance to use the different tools and controls related to the mill.

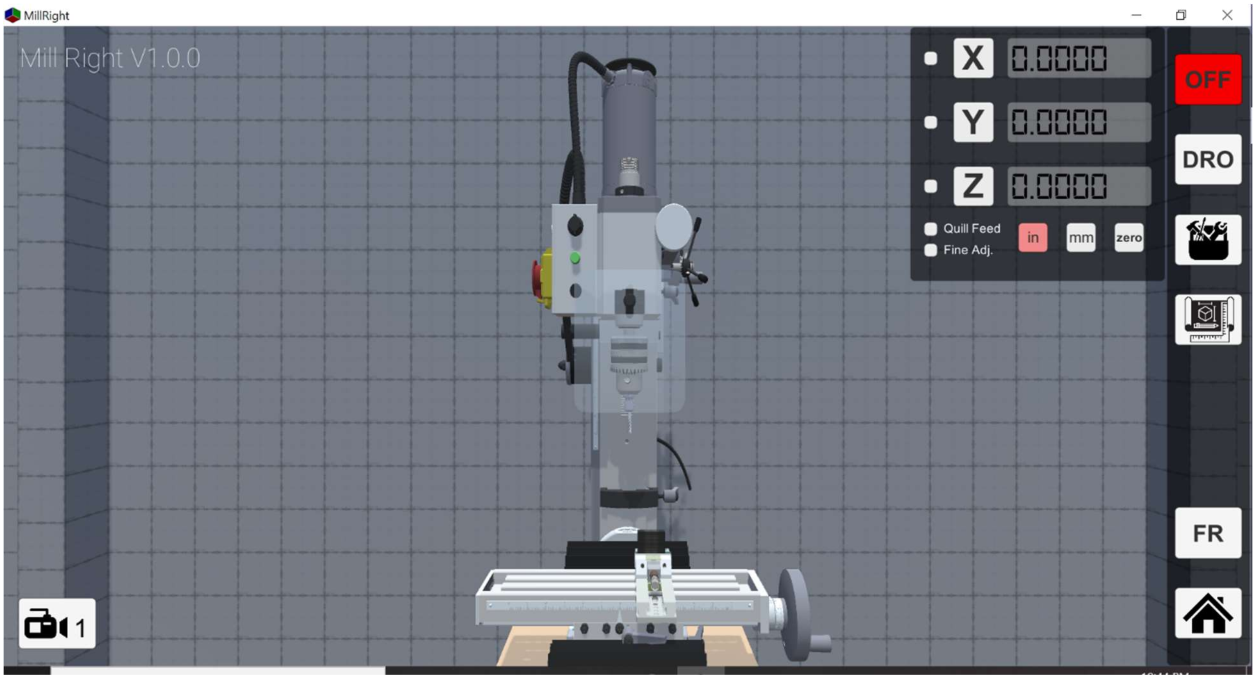


Figure 2.6 Mill Simulator Scene (Milling Simulator Module) display of initial setup after clicking DRO panel

3 How it's made

3.1 GameObjects, Scenes, and other 3D Assets

3.1.1 3D Models

Most of the 3D models included in the Mill Right simulator are assets that have been imported from GrabCAD community libraries. These assets include the mill [1], vise [2], workbench [3], endmills [4] [5] [6], drill bits [7] [8] [9], drill chuck [10], and wrenches [11]. All assets retrieved from GradCAD are free to use according to the GradCAD Website Terms of Use [12].

3.1.2 Camera System

Each scene uses a collection of camera GOs which allow the user to toggle between various camera perspectives with the click of a button. Each camera GO contains it's default "Transform" and "Camera" components. The "Transfrom" component stores the position, rotation, and scale of any GO. The cameras are children of the **Mill** GO so that the cameras local position is always relative the **Mill**. This allowed the Mill to be imported into any new scene during the development process without the need to reposition each camera for that specific scene.

3.1.3 User Interface & Controls

Each scene has a user interface (UI) that is slightly different from the UI of other scenes. However, each of the UIs share the same general structure depicted in Figure 3.1. Eeach UI begins with the UI_Panel GO at the top of its hierarchy. The UI_Panel is a GO containing the default components that are instantiated when creating a new Canvas GO in Unity's Hierarchy window. These default components include the "Canvas", "Canvas Scaler" and "Graphic Raycaster". Generally speaking, these default components serve to render the various elements of the UI overlaid on the screen space and allow those elements to be interactable. In addition to the default components, the UI_Panel also carries a C# script, Camera_Toggle.cs, that allows the user to toggle through different camera perspectives in a scene.

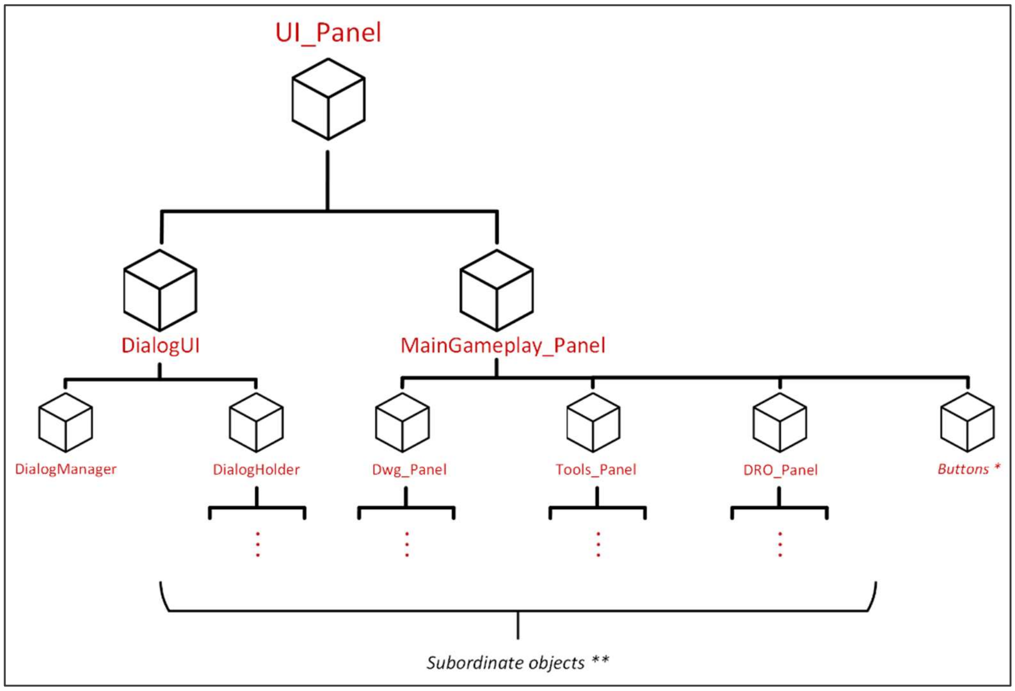


Figure 3.1 Hierarchy outlining the different GameObjects that make up the user interface. (*) denotes many distinct button objects. (**) denotes many GameObjects of various types

The DRO_Panel, Tools_Panel, and Dwg_Panel share similarities in basic aspects of their design. Each contain a "Canvas Renderer" component (to render the panel on screen when attached to a Canvas), and an "Image" component for the panel backdrop. Unique aspects of their respective designs are considered in the following sections.

3.1.3.1 Digital Readout (DRO) Panel

The DRO_Panel is used to facilitate user control of the mill, as well as display the spatial positioning of the cutting tool relative to a point of reference. Each plane of motion (x, y, z) has it's own reference point corresponding to the "Transform" component of an object within the scope of a given Animator. This implementation was necessary to access the local position of the reference point, which is relative to it's parent that is controlled by the Animator. More on the relevant animators in section 3.1.5.1.

3.1.3.2 Drawing (Dwg) Panel

The drawing panel contains 2 GOs that each hold an “Image” component. 1 image component contains a reference to the workpiece schematic sprite in metric units, the other image contains a reference to the version of the schematic with imperial units. To create sprites for the panels shown in Figure 3.2, the original AutoCAD .dwg files were exported as .png files, then subsequently imported to Unity’s asset menu. Afterward, the texture type of the newly imported asset must be set to “Sprite (2D and UI)” via the Inspector window.

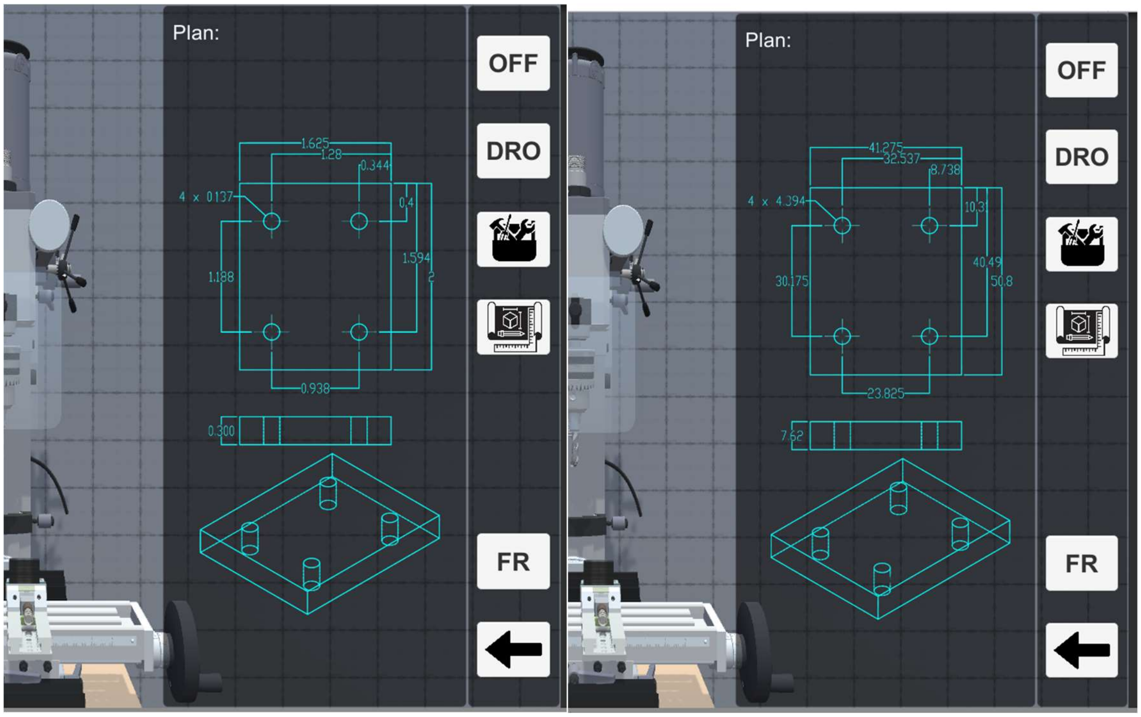


Figure 3.2 The Dwg_Panel shown with both metric and imperial versions of the workpiece schematic

3.1.3.3 Buttons

The DRO_Panel, Tools_Panel, and Dwg_Panel share similarities in basic aspects of their design. Each contain a “Canvas Renderer” component (to render the panel on screen when attached to a Canvas), and an “Image” component for the panel backdrop. More design considerations specific to each of these panels will be disclosed in subsequent sections.

3.1.4 Stock Material

The stock material is a GO which is composed of a set of cubes and a Particle System. The particle system is a technique in game development that uses many 3D models or sprites to simulate certain phenomenon (which would be difficult to accomplish with standard rendering techniques). This system was used to simulate material being taken off the stock material. The cubes inside the stock material are standard unity 3D cube meshes that possess “Collision Visibility” script and a box collider. The “CollisionVisibility.cs” script enables an object with the correct tag (Collision tag) to start the collision process. This process involves the particle system being set active and transported to the location of the cube and the cube is destroyed after a set amount of time. If at any time during the process the collision ends before the cube is destroyed, it simply resets the countdown and deactivates the particle system. The stock material is designed to allow the user to perform all the basic types of operations possible with a milling machine.

3.1.5 Animations

The simulator is based on the use and control of animations. Animations clip were used to control the different axes of the mill and their respective handles. When a user interacts with the

DRO panel, it triggers the start of an animation that stops as soon as the interaction ends (scrolling the mouse wheel or touchpad). The scripts related to performing this feat will be discussed in subsequent sections. The animations were done by using animations clips and animators. An animation clip stores keyframe based animations. It is one of the cores elements to animation system found in Unity. The animator is the interface used to control the animations clips.

3.1.5.1 Milling Animations

For the mill operations, the user needed control as to be able to play forward and rewind. The animators related to the control of the mill all have a parameter “Reverse”. This parameter is used as a multiplier and is set -1 when the user scrolls in the opposite direction. This allows the user to control the mill in both directions. The animations related to the mill can be found in the Mill Animation folder.

3.1.5.2 Tool Setup Animations

The tool setup animations are controlled using pause and play. Those animation clips are controlled using an animator that performs the animation on loop at runtime. The animator automatically transitions between the animation clips after their completed. The animations related to the tool setup can be found in the Tool Setup Animation folder.

3.1.5.3 Vice Animations

The vice animations were built but not implemented. These animations were to be implemented with the intention of letting the user control the stock material and the vice to perform freely all different types of operations. The control for this feature would involve a UI slider. More information of this topic will be included in further developments.

3.2 Scripts

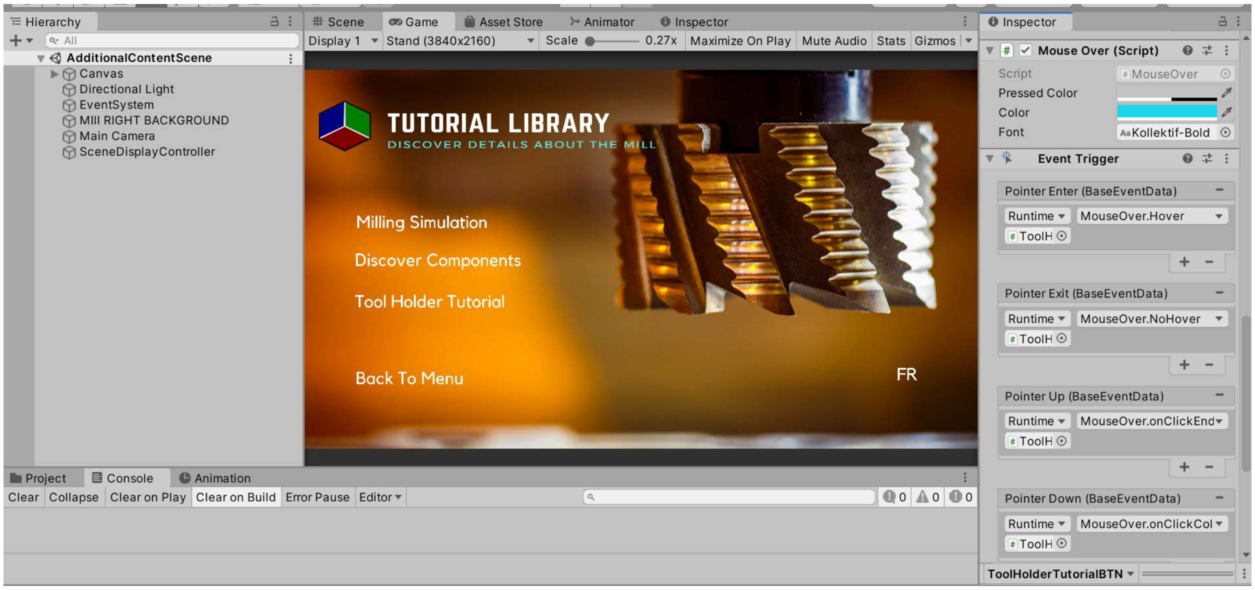
Unity allows users to create their very own components using scripts. These scripts can trigger events, modify components property, and respond to user important. Scripts can be described as the core of the simulator.

3.2.1 Dialogue Scripts

The dialogue scripts were used to handle all events related to the dialogues (see figure 2.4 & 2.5 for dialogue examples). The script "Dialogue.cs" is a class which holds the name of the dialogue (string) and an array of string (sentences) which are the dialogues. The dialogue class includes an array of sprites to put relevant images and it also features the translation of the sentences and name for accessibility purposes. The entire class is made a "System.Serializable" to allow the class to show up in the inspector and edit its content. The script "DialogBox.cs" is a "ScriptableObject" which includes only a List of the class Dialogue. A "ScriptableObject" is a data container that can collect large amounts of data (separate of class instances). The data that contains the dialogues can be found in the Resources Folder, inside the content folder under the name "DialogsIntro" (scriptable object is editable if any change is necessary). These scripts were the base of the dialog box implementation. To control the scripts, it was necessary to implement scripts to trigger and manager the dialogues. The "DialogueTrigger.cs" script has access to the "DialogueBox.cs" script and triggers the start of a new dialogue from the list in the "DialogsIntro" scriptable given an integer (index). The Dialogue is sent to the manager (“DialogueManager.cs”) and the manager executes the logic to switch to the new dialogue.

3.2.2 Main Menu Scripts

The main menu scripts involve the logic necessary to user interface aesthetics. The “MouseOver.cs” implements the logic relating to the color and font when a button is clicked or hovered over. The functions of this script are called when events triggered by the GO component “EventTrigger” are called. For instance, on “Pointer Enter” (mouse is hovering over button), the “EventTrigger” calls the method “Hover” from “MouseOver.cs” which changes the color and font (see figure 3.2 for trigger events of “Tool Holder Tutorial” button).



Mouse Over (Script)
Script: MouseOver
Pressed Color
Color
Font: Kollektif-Bold

Event Trigger

Pointer Enter (BaseEventData)
Runtime: MouseOver.Hover
ToolH

Pointer Exit (BaseEventData)
Runtime: MouseOver.NoHover
ToolH

Pointer Up (BaseEventData)
Runtime: MouseOver.onClickEnd
ToolH

Pointer Down (BaseEventData)
Runtime: MouseOver.onClickCol
ToolH

Figure 3.3 Additional Content Scene Editor (TriggerEvents of Tool Holder Tutorial Button)

3.2.3 Mill Control Scripts

The mill control scripts are the core of the essentially controlling the mill. These scripts include controls for each possible manner of mill movement. Further implementation would include creating a parent class to improve performance. The movement in each direction is implemented with the help of animations clips. The scripts “XWheelControl.cs”, “YWheelControl.cs” and “ZWheelControl.cs” all control animation clips. At runtime, each control scripts sets the speed of the animation to 0 (using the method “pause ()”). When the necessary steps are taken to enable their movement, the data from user’s mouse scroll (or touch pad scroll) is sent to the respective control script. If the information is to control the script in the forward direction, the parameter “Reverse” (discussed in section 3.1.5.1 Milling Animations) is set to 1 and the speed of the animation is set to 0.05f (using method “setSpeed(float argument)”). To control the script in the opposite direction, the “Reverse” parameter is set to -1 and the speed is set in the same manner explained latterly. The “QuillFeedControl.cs” and “FineAdjustment.cs” are like the axes control scripts but they differ in the movement of their parts. To interact with the stock material, it was necessary that these controls were done changing the local position of the quill feed in the mill instead of using an animation. The animation was still used to control the quill feed handle and the fine adjustment knob. Furthermore, to avoid drill bits or end mills going inside the stock material, the scripts possess a Boolean (collision variable) that stops the animation/movement in the specific direction is the collision variable is true. The “GameButtonManager.cs” is the script that turns the mill ON/OFF. It starts the rotation animation of the bit and enables the interaction of the variable “currentBit” with the stock material by setting the tag on that GO to the variable drillTag. As outlined in section 3.1.4 if the GO with the correct tag is colliding with the stock material, the destruction process starts. The drillTag in

“GameButtonManager.cs” is the same as “collisionTag” in the “CollsionVisibility.cs” script. When it is off, the tag is switched back to the GO’s original tag and the animation stops. Finally, the “ControlManagerScript.cs” simply makes sure that the user cannot interact with the different controls concurrently.

3.2.4 Mill Scene Scripts

The mill scene scripts are used to setup the hover events and on click events relating to the mill. In the same format as the “Dialogue” class, the mill scene script has a “ComponentDetail.cs” class which is used by a ScriptableObject (“PartDataScriptable.cs”) to store the data of each component of the mill. The “ComponentManager.cs” script is used to handle the logic related to switching between different components on click. The “SetupOnHover” essentially sets up it’s children GO with an “OnHover.cs” script and the necessary details to interact with the “ComponentMangager.cs” and call the method “SetDetails(int index)” with the value “detailIndex” (integer in “OnHover.cs” script). The “OnHover.cs” script also possess the logic to change the color of the material when the mouse pointer hovers on the mesh. This feature was added to show which part of the mill is interactable. The method “SetDetails(int index)” essentially takes the index and switches the current instance of “ComponentDetail.cs” with the instance at the index in the list “List<ComponentDetail> details” (Scriptable can be found in Resources/Content/ComponentDetails.asset). The information of the new component detail is passed onto the description panel and shows the details of the part clicked (see figure 2.2 for example).

3.2.5 Spindle Scene Scripts

The spindle scene scripts are used by the logic behind UI Panel found in the Spindle Scene (Tool Holder Setup Module). Similarly, to the mill scene, the spindle scene uses a list of component details built using a scriptable object (Scriptable can be found in Resources/Content/ AddtionnalInformation.asset). The UI Panel possess buttons that call the method "setDetails(int index)" from "AddtionnalInfoManager.cs" (logic works the same as the "ComponentMangager.cs") and sends their respective index to prompt details about the information contained "ComponentDetail.cs" instance in question in the list. The "PanelToggle.cs" script simply toggles between the explanation of the tool holder setup and the description panel (which explains the different parts involved in the tool holder setup).

3.2.6 Stock Material Interaction Scripts

The stock material scripts are the core of the execution of the interaction between the different milling tools and the stock material. The "SwitchBit.cs" script is what enables the user to switch between different drill bits and end mills (as well as their tool holders). The "StaticRigidBody.cs" is the script which hold the different drill bits and end mills in place and changes the value of different collision variable for the Mill Control scripts. The "CollisionVisibility.cs" script works as outlined in Section 3.1.4

3.2.7 Toggle Scripts

The toggle scripts are used to toggle between different languages, images and game objects. The “Camera_Toggle.cs” is the script which enables the different camera perspectives seen in the different tutorial scenes. The “ToggleText.cs” is used mostly to change between French and English. The “ImageToggle.cs” is used for implementation like the “Show” and “Hide” sprites which change on click.

3.2.8 UI Panel Scripts

The UI Panel Scripts handle most if not all the logic relating to the UI Panel. It handles the interaction between the DRO Panel and the control scripts. It also handles linking the Tool panel with the “SwitchBit.cs” script. The UI Panel enables the user to control different aspects of the different modules. It is essentially the user’s game controller.

3.2.9 Scene Management Scripts

The scene management scripts handle the logic relating to the transition between the scenes. Furthermore, to pass the language switching, it was necessary to create a variable which keeps track of the value of the language and change the language according to the value of Boolean. That variable can be found in the script “LanguageSceneToggle.cs”. In a similar manner a “ButtonSceneToggle.cs” script was made to control the toggle of buttons that transitions to the different scenes depending on where it is accessed from. When the Mill Scene is accessed using “Begin Tutorial” button, the next button is activated, and the back button is deactivated. However, if the Mill Scene is accessed from the Additional Content Scene, the back button is activated, and the next button is deactivated. The “LanguageSceneSwitcher.cs” handles the logic of switching the language for all the different GO at when the scene is prompted at runtime. “LoadScene.cs” simply switches the scene to the scene which name was put in the in “loadlevel(string scene)” method. This method is called with the help of buttons on click.

4 Installation, Operation, and Maintenance

4.1 Installation & Operation

To build the game, it is important to make sure that all the scenes are included in the “Scenes in Build” list. Furthermore, if any scripts are using Unity’s editor, please remove them or place them in the Unity Editor Folder (create one if not present). Finally, select the directory to build the file and click build. This will create the executable file. To perform any icon changes, or product name changes just access the player settings located in the build settings. In the Player Settings, it is possible to change the Splash Screen.

4.2 Maintenance

When performing maintenance and updates, please look at section 4.1 before building game. If the maintenance is in relation to content, please access Resources Folders which hold the important content of the simulator. If the maintenance is in terms of adding content (performing a patch update), please make sure to understand the different scripts already implemented to avoid troubleshooting. A description of the logic of the different scripts is available in section 3.2 and the scripts are divided in different folder (divided into folders with similar purposes) to allow easy access and updates.

5 Health & Safety Precautions

There are no health and safety precautions that are relevant to the use of this specific product. Instead, consumers are urged to follow the precautions outlined in the hazard report for computer ergonomics provided by the Ontario Ministry of Labour [13].

6 Troubleshooting

6.1 Animation Issues

The main problems that we ran into in the creation of the simulator was controlling the mill. Animations are very hard to control and because of the time constraints the implementation for the control of those animations is not ideal. As such, we ran into debugging issues because a lot of scripts had access to the variable inside the mill control scripts. Furthermore, Unity does not automatically change the name or directory of the GO in the animation tab when it is changed in the hierarchy tab which led to a lot of problems in terms of the animation. For instance, if the lock handle changed directory, when the animation is running at runtime, it would be left behind. It was thus difficult and tedious to change both the GO in the hierarchy and in the animation tab.

6.2 GitHub

Throughout this project it became evident that while GitHub is a great version control software, it is not ideal for Unity. In the initial stages of the project, the group ran into a lot of merge conflicts due to editing the same scene. To remedy this, it was necessary to avoid editing the same scene at the same time as it would create a lot of merge conflicts.

7 Recommendations for Future Work

7.1 Unity Teams

For future updates and works relating this project, it would be ideal to use Unity's Version Control (Unity Teams) because even though there's a cost relating to this version control, it would allow for better teamwork and avoid merge conflicts.

7.2 Safety Content

Due to time constraints, this content could not be added but it was going to be part of the simulator. The implementation included warning and safety dialog boxes when user makes a mistake. It is a necessary content to add to future work because the mill should be safely and efficiently. This simulator's goal is to teach users how to do use the mill safely and efficiently.

7.3 Vice Control

The vice control is a feature that the team wanted to add to allow the user to place the vice anywhere on the table and change the stock material placement on the vice. This feature was to be added in the Milling Simulator Scene for the purpose of allowing the user complete freedom in operating the mill and performing different operations.

8 Conclusion

To conclude, the user manual documented the different tutorial modules, how the final prototype was constructed, the operation and maintenance of the latter, the precautions needed to use the product, the main issues the group ran into while creating the product and the recommendations that would be ideal for future patch updates of the simulator. All the different models used in the production of the simulator can be found using the references detailed in section 9. This user manual was detailed with the goal of guiding users into how the product was made in order for them to take the mantle and further bring along this project.

9 References

[1] Mat, "Mini Mill CX605," 11 August 2015. [Online]. Available: grabcad.com/library/mini-mill-cx605-1. [Accessed June 2020].

[2] Maury, "Milling Machine Vise," 22 February 2015. [Online]. Available: grabcad.com/library/milling-machine-vise-1. [Accessed June 2020].

[3] technovo.cz, "Kosan XL Workbench," 14 July 2019. [Online]. Available: grabcad.com/library/kosan-xl-workbench-1. [Accessed June 2020].

[4] S. Slanda, "3 Flute Endmill to Aluminum," 26 October 2015. [Online]. Available: grabcad.com/library/3-flute-endmill-to-aluminium-1. [Accessed June 2020].

[5] S. Slanda, "4 Flute Flat Endmill," 26 October 2015. [Online]. Available: grabcad.com/library/4-flute-flat-endmill-1. [Accessed June 2020].

[6] S. Slanda, "6 Flute End Mill Cutter," 30 October 2015. [Online]. Available: grabcad.com/library/6-flute-endmill-cutter-1. [Accessed June 2020].

[7] m. r. hossain, "Drill Bit," 19 June 2020. [Online]. Available: grabcad.com/library/drill-bit-165. [Accessed June 2020].

[8] D. Petrut, "A drill created in Solidworks," 18 June 2020. [Online]. Available: grabcad.com/library/a-drill-created-in-solidworks-1. [Accessed June 2020].

[9] M. Hejjas, "This is how a real drill bit looks like," 16 January 2019. [Online]. Available: grabcad.com/library/this-is-how-a-real-drill-bit-looks-like-1. [Accessed June 2020].

[10] U. Lothwer, "Drill Chuck," 19 March 2020. [Online]. Available: grabcad.com/library/drill-chuck-11. [Accessed June 2020].

[11] J. Romero, "Wrench Set (Juegos de llaves)," 8 July 2020. [Online]. Available: grabcad.com/library/wrench-set-juegos-de-llaves-1. [Accessed July 2020].

[12] Stratasys, "GrabCAD Website Terms of Use," 28 June 2016. [Online]. [Accessed June 2020].

[13] T. a. S. D. Ontario Ministry of Labour, "Rest Breaks for Computer Operators - Health and Safety Guideline," The Government of Ontario, February 2011. [Online]. Available: www.labour.gov.on.ca/english/hs/pubs/gl_restbreaks. [Accessed July 2020].