CHILI
computer-human interaction
in learning and instruction

LEARN
Center for Learning Sciences

Design of a human interaction activity with Cellulo-MORI

Author:
Antoine Clivaz

Supervisor(s):
Hala Khodr
Kevin Holdcroft
Barbara Bruno

Professor:
Pierre Dillenbourg

January 7, 2022

Contents

1 Introduction

The purpose of this work was to design a learning activity on emergence using *Cellulo*s and *Mori*s, robots either developed at CHILI[1] or RRL[2]. The **emergence** is the ability of multiple elements to develop a property or an order that they don't have on their own (e.g. the organisation of an ant colony or arrangement of atoms in a solid).

Altogether, we seeked to create a simulation which lets the user discover what the emergence is by using our robots: to have a common goal that you can only achieve by joint efforts of all robots or users which are taking part in the simulation.

This project and the project of Jean-Etienne Charbonnet were done simultaneously and are part of the *Grass Root project*, as it will be explained in the section below.

2 General background and expectations

First of all, let us recall what was at stake and why this project was born. According to the **Grass Root proposal**, the main goal of this research is to provide a validation on a theory based on the idea of Piaget, Montessori and Frobel that hands-on and physical manipulation have lots of benefits in learning and education.

Using *Mori*s, that are self-reconfigurable modular robots (SRMR) and the educational *Cellulo* robots, we would be able to create a learning activity that allows users to perform local actions by using the *Cellulo*s and concurrently see their global effects through the *Mori*s 3D topologies, thus grasping the concept of emergent behaviors.

Before going further, here are representations of the robots that we will use for this work. From left to right: a *Cellulo*, a *Mori* and a *Cellulo-Mori* (CM). The *Cellulo-Mori* is constituted of a *Cellulo* as a base, which gives us some abilities. For instance: moving, communicating with others and displaying colors on LED's. A *Mori* is attached on top of the *Cellulo-Mori*, which gives us the ability to connect with other robots (*Mori*s and *Cellulo-Moris*).

Each robot has specific properties that bring a lot to the activity: *Cellulo*s were designed as robots for education and learning, thus have several components that can be used to make visible and tangible what is invisible and intangible. This provides us with haptic feedback to interact with the user, motors allowing us to move the robot in a plane, LED's for visual feedback, etc.

[1]Computer-Human Interaction in Learning and Instruction: `https://www.epfl.ch/labs/chili/fr/chili/`

[2]Reconfigurable Robotics Lab: `https://www.epfl.ch/labs/rrl/`

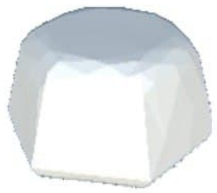


Figure 1: Cellulo

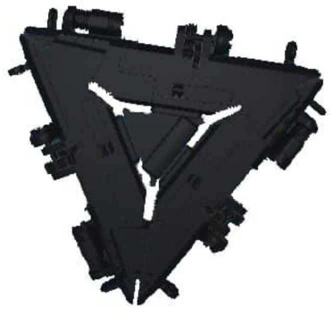


Figure 2: Mori

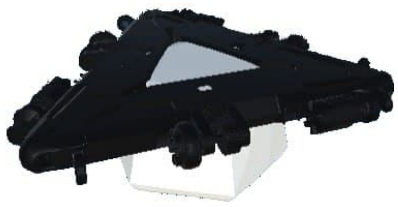


Figure 3: Cellulo-Mori

On the other hand, *Mori*s are *SRMR* robots i.e. modular robots that can take different shapes by connecting with others. This allows us to interact with the 3D world and brings a new dynamic to the activity as the *Cellulo*s were bound to only act in a horizontal plane.

A combination of those robots provides us with a really interesting tool for hands-on and physical manipulations. Secondly, recall that this project is part of another one: the Grass Root project. What were our goals and expectations?

The *Grass Root* project had three main goals: redesign the *Cellulo* robots for modularity; integrate *Mori* robot as an interaction layer; design and validate an activity on emergent behaviours that relies on *Cellulo* and *Cellulo-Mori* robots.

In other words, this work and the one of Jean-Etienne Charbonnet are bound together and take place as the third goal of the Grass Root project. Our objectives were hence to find an idea of a game that meets the third goal's objective and to implement it.

3 Getting started

3.1 Unity

As the goal of this project is to develop an activity/simulation in Unity, the first task was to get familiar with Unity and all its features. Indeed, Unity is a 2D/3D game engine and cross-platform IDE (Integrated Development Environment) that provides most of the features needed to make games work properly, for instance physics, collision detection or even 3D rendering. With Unity, no need to start a project by defining a physics engine from scratch or to implement how the light should reflect on surfaces. This can be done using already built-in features, making Unity a great choice for the project. Furthermore, Unity IDE provides a visual editor that allows to simply drag and drop items into scenes as well as directly adjusting their properties. Although Unity allows to implement a lot without having much code, the programming language used to handle code and logic is C# which can be seen as a hybrid that takes the best of C and C++.

3.2 Baseline

At the beginning, we were given a base project with already imported *Cellulos*, *Moris* and *Cellulo-Moris* robots prefabs, as well as already implemented features such as moving the robots around or moving joints between two connected *Moris*. We were also provided a base scene that appeared to be very useful to understand the basics of Unity and explore the possibilities it offers to develop the activity, see Figure 4. After about two weeks of learning C and Unity, we had to split the work into two main parts, the Physics and the User Interface development. Jean-Etienne took care of the Physics as I took care of the UI, so that we achieve a complementary work and do not work twice on the same aspect of the project, while we keep meeting every week to discuss and share our own advancement with the team to keep on track.

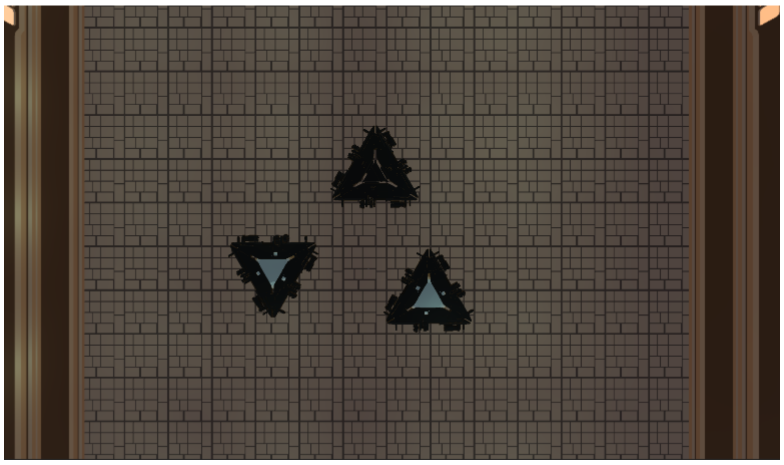


Figure 4: Base Project Scene

4 Features

Before diving in the activity design and development, my first task was to implement features that can then be used in the said activity. Here are some of the features developed and used in for the creation of the final activity:

4.1 Mori Spawner

If a *Cellulo-Mori* wants to connect to a *Mori*, the *Mori* in question must lay at the same height as the *Mori* that is on top of the *Cellulo*. It can also be interesting to have the option to grab several *Moris* from a dedicated place. The *Mori Spawner* is a small platform that acts as a dispenser, allowing to directly grab a *Mori* at the perfect height. Also, by left clicking on it, the user can spawn a *Mori* on top of it.

4.2 Robot spawn feature/UI

Having a *Mori Spawner* is useful, but what about being able to spawn any desired robot on a specific spot. I implemented a set of buttons that on selection, enables the user to make the selected type of robot appear on the map by left clicking. He can then move it around as well as rotating it using the mouse wheel. When the user is happy with the position of the robot, he can make it spawn by left clicking again. Robots can also be removed by the user, by right clicking on them.

4.3 Cameras

In addition to the top view camera that was provided in the baseline, I added a third person view camera that is following the selected robot, making the activity more immersive and offering a second view option for the user. For this task, I use Cinemanchine, a suite of tools for dynamic, smart and codeless cameras. It offers a wide range of possibilities, including a FreeLook component that can make a camera follow a given gameObject in real time with multiple options of view field and a Collider component making possible, for example, the camera to collide with walls instead of going through them.

4.4 Color travel (Connection Graph)

As *Cellulos* and *Moris* are equipped with color LEDs, an interesting feature is to make a color travelling from a robot to another one in a connected cluster of robots. This can be used when designing the activity as an item exchange between two *Cellulo-Moris* through a bridge of *Moris* for example. To get a good view of every cluster of connected robots, we use a generic graph and to get the route the color will have to take, we use DFS (Depth-First Search). Each robot is given a unique ID at spawn and an edge is added/removed to the connection graph whenever a connection between two robots is created/destroyed. The generated connection graph can be further used to another purpose than finding a path for a color to travel from a robot to another.

5 Activity Design Process

Throughout the semester, we had to think of activity ideas according to several points. There are 3 types of robots available, each having its own strengths and weaknesses. At this point one big challenge was to identify and make the most out of them. We also had to keep in mind that, as the simulation is at an early stage of development, there are features that can already be done in the real world but that would be hard to implement besides the activity itself (e.g., mouse/keyboard control key binds of multiples joints between *Moris* in a cluster of connected robots).

5.1 Type of activity

To define and develop the final activity, we had to go through a design process that started with brainstorming game ideas according to multiple criteria. For instance, have multiple robots acting, have a human interaction, use *Cellulo-Mori* capabilities, let player have multiples options and so on. Here are some ideas we came up with:

- **Deliver Game**: Simple task of bringing a robot, a color or an item carried by robots from point A to point B.
- **Obstacle Game**: A robot faces diverse obstacles and the player has to find the best way to overcome them with the resources he has at disposal. These obstacles can take the form of a fence to go over or a river to cross for example. See what they could look like in Figure 8 in Appendix. At this stage of the project, the control of connected robot in the simulation is not very handy, it hence limits the panel of obstacles that would require precise simultaneous movement of robots or connection joints. It was also important to think of how an obstacle can be crossed and what we do not want the player to do. Limiting the ways to overcome obstacles while having the player think which option is the best.
- **Hole in the Wall**: Inspired by the *Hole in the Wall*[3] game show where the player has make his body have the same shape as the hole in a wall sliding towards him to get through it. A *Cellulo-Mori* would have to grab given *Moris* to form a shape that matches the one in the wall that would come towards it after some time. This could be played in coop with two *Cellulo-Moris* having to collaborate to create a bridge of *Moris* of a certain shape between them. This game could be very interesting to have player in the real world think of the best shape in a 3D plane that would go through a wall that is in 2D and build it. But unfortunately, it is limited to play such a game only in the simulation with the resources and 3D controls handiness we have.

[3]Hole in the Wall game show: `https://m.imdb.com/title/tt1277979/mediaviewer/rm3433206784/`

- **Maze Game**: Have a robot progress in a maze where it will have tasks to achieve in order to get out of it. For example, grab a *Mori* and bring it to a defined spot to unlock a door or spawn another useful robot. This game could be thought as a coop game where two robots would have to collaborate to get both out of the maze. It could also be adjusted to a mirror maze game, where the player controls a robot that is outside the maze and has his action be duplicated on the robot inside the maze but in mirror.
- **Wolf, Goat and Cabbage Game**: Create a version of the well known problem[4] with our robots. One person controls a *Cellulo-Mori* as a boat and the other can connect the actors represented as *Mori*s (wolf, goat, cabbage) to the boat in order to make them cross the river. One person controls a *Cellulo-Mori* as a boat and the other can connect the actors represented as *Mori*s (wolf, goat, cabbage) to the boat in order to make them cross the river.

All these ideas had some great components but also limitations that we had to keep in mind. They could also be customized by adding some limitations to the control features. For example, allow certain robot to move along only one axis or disabling their ability to rotate. Game ideas could also be mixed to add more substance to the activity like having a maze where you control a *Cellulo-Mori* and have to collect *Moris* in order to form a shape that fits a hole in a wall, opening the exit door. While thinking of new ideas every week, I developed a small game where the player had to collect gems on a map to unlock the final star, see Figure 5. It helped me implement basic game rules and winning conditions that could be used later for the final activity.

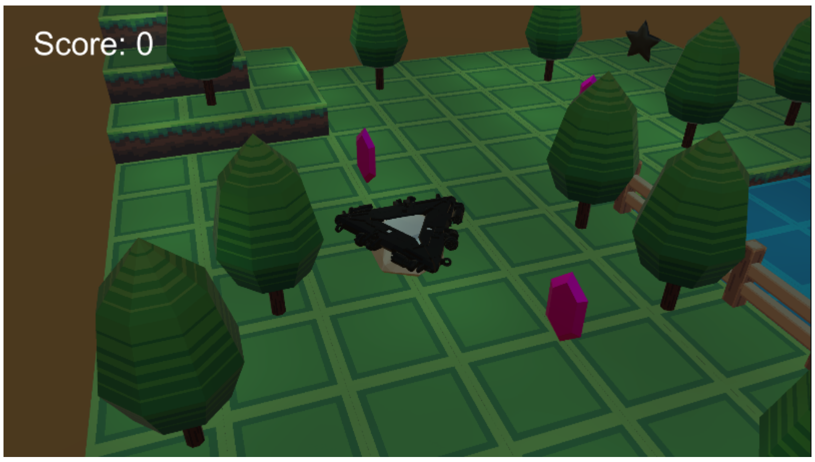


Figure 5: Basic Collect Game

[4]`https://en.wikipedia.org/wiki/Wolf,_goat_and_cabbage_problem`

5.2 Final Activity Design

After having analyzed different activity options, available resources and time left for implementation, I decided to merge several above cited game ideas. The basic goal of the activity will be to bring a color, that starts on a robot, at point A to a defined end point B. Furthermore, the activity will take place in a maze where the player has to complete tasks and overcome obstacles to open doors or progress in the game. As we did not have then a proper way to control multiple connected robots in the simulation, the tasks and obstacles should remains quite simple and should not require the use of too many robots. After some trials, I ended up with a draft version of what the final mate could look like, see Figure 6. A color is trapped on an island and requires to be brought to the final rectangle zone. A *Cellulo-Mori* is trapped in a small prison and can be freed by connecting and collaborating with the other *Cellulo-Mori* and the *Mori* to collect gems. The color needs a bridge to pass across the river. Multiple mechanics involving collaboration and connections between robots will lead the player to the final zone and activity achievement.

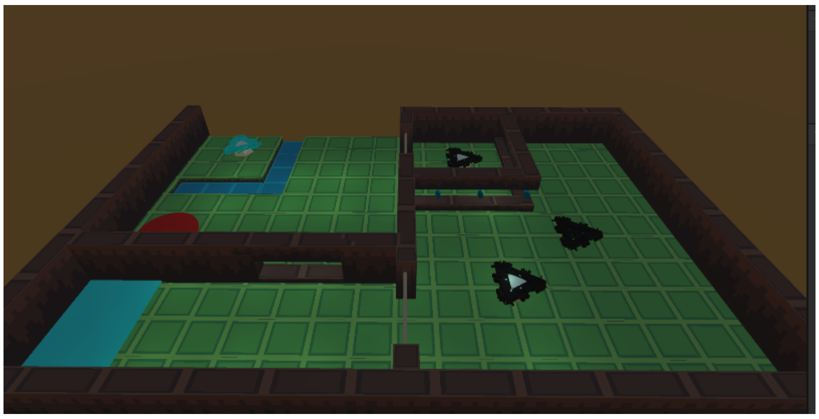


Figure 6: Draft Maze Activity

Each mechanic and trigger like changing color, opening doors, showing the end screen and so on, is handled in small scripts that all together make the game work. All the connections between robots were done manually so far, but Jean-Etienne came up with the great feature that allows robots to connect to each other automatically[5]. From now on, I had to keep improving the maze activity and some features of the robots while keeping it compatible with Jean-Etienne's work that involved NashMesh components for example.

[5]See Jean-Etienne's work for more on information on self alignment and connection.

Encountered problems

- During the activity robots with a NavMeshAgent component can only navigate on a NavMesh area that is created beforehand. This walkable area is defined according to a selection of layers we want the robot to be able to navigate on. As the maze is composed of doors that we want the robot to be able to go through when they are open, we create the NavMesh area so that these doors are walkable. However, if a robot tries to go through a closed door, its body will be blocked due to its RigidBody but its NavMeshAgent will go through the door, shifting its center from the center of the robot and creating unwanted behaviour. This is solved by adding a box shaped NavMeshObstacle component on the doors that will prevent all robots' components from goind through a door.
- The prison obstacle initially required having two *Cellulo-Moris* connected together via a *Mori* to collect the gems, but as we did not have the time to implement a feature that allows a cluster of connected robots to move properly, I had to rethink this obstacle. I ended up with a bit different prison that now requires the *Cellulo-Mori* outside to grab a *Mori* and collect two gems that are position at a height, while the trapped *Cellulo-Mori* simply has to collect gems inside the prison. When all the gems are collected, the door opens. This modification still involves two robots and a connection with a *Mori*, but looses some of its collaboration aspect.

As the visual appearance of a game is important for the player to have the best experience possible. I created a similar maze with a new design that looks cleaner and more sober. To make the game look even nicer, from imported packages, I added trees, a small hill and some visual effect like a magic looking circle and an end zone with flying particles. I also put visuals (a magic circle and a gem) on the doors to make it clear what the player has to do. See Figure 7.

In such an activity, it is interesting to have a score metric to assess how well a player performed. It can be the number of robots used, the number of connected components or the time elapsed until the completion of the game. I chose to use the time as score metric, the fastest one finishes the game, the higher the score he will get.

The activity itself is concentrated in only one scene and there is no real instruction on what the player has to do nor the controls to use to move a robot or change camera view. Therefore, we created a main menu on which the player will land when launching the activity. It allows the player to start playing, get access to the game instruction and controls (keybinds) or quit the application. A pause menu was also created for the player to get access to the controls if he needs a reminderor to go back to the main menu if he wants to restart the game. As there is a lot of options and features to control the robots, a tutorial is available for new players to get guided and have an overview of what can be done in the game.

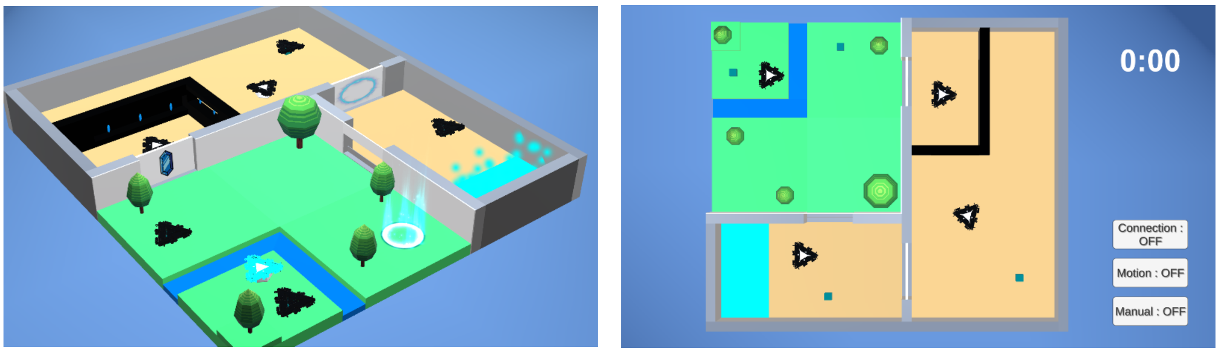


Figure 7: Final Maze Activity

One aspect of the project would have been to *preliminary validate the learning activity on emergent behaviours that relies on Cellulo and Cellulo-Mori robots*, but due to lack of time, we did not have the chance to have users test our activity and get feedback on it. A questionnaire[6] was prepared but the game was not ready to be tested as some bugs remained.

6 Possible Upgrades

- Implement a feature that allows the user to control a cluster of connected robots. How they rotate together. What is the center of the cluster and can it be changed. If the user can control a whole cluster as one unit, it unlocks mechanics and possibilities for future activities.
- Improve the way joints between *Moris* are controlled. This would facilitate the creation of more complicated obstacles and new game features.
- Have a reliable way to control robots in the simulation via the robots in the real world. As it is hard to control multiple joints at the same time only using controls in the simulation, it can ease some manipulation and manoeuvre working with the actual robots. We did not have the chance to work with the physical *Cellulo-Mori* during the project as it was still under development but hopefully it will be soon possible.
- Add check points to the game so that if something unexpected happens, the player can try again without having to start from the beginning.

[6]Google form on activity feedback: `https://forms.gle/2x1BJAQHA5QpQ7zX8`

7 Conclusion

Throughout the project, I learned a lot about what was feasible using Unity and how easy some complicated elements could be implemented using built-in features as well as how difficult some trivial elements were hard to implement. Having prior knowledge in Unity and C would have made the project easier and code cleaner. Tasks can be done in various ways in unity, you can either have one big script that handles multiples features, have numerous small script that each take care of some features or even use options in the unity inspector to take care of some elements. For a beginner in unity, it is hard to sense which option is the best in each situation. We did our best to have an activity that is playable. You can find the code on Github[7].

Working on an ongoing project was very interesting and exciting as we had the chance to have weekly meeting with the principal actors of the project. Having the chance to expose ideas to the team and get positive feedback as well as further ideas made the developing process really enjoyable and fulfilling. The activity design process was surprisingly riveting and was great to experience in an amazing team.

Working with Jean-Etienne on the project allowed me to have a broader view and understanding of aspect I did not directly work on. Even though we had different roles on the project and were not really working together, it was really helpful to have someone I could rely on. We had some issues to merge our work as unity provide files that are not really suitable to merging option on c4science and it made the merging process more tedious than expected.

To conclude, this project gives us an insight of what kind of activity involving *Cellulo*, *Moris* and *Cellulo-Moris* can be developed in Unity. It also shows us that Unity is a very handy tool to use when making a game but has its own limitations for real physics simulations. More options can still be explored to get the best and most realistic simulation using Unity.

[7]`https://github.com/DeepGreen1/cellulo-mori/`

Acknowledgments
I would like to sincerely thank Hala Khodr, Kevin Holdcroft and Barbara Bruno for their availability and guidance throughout the semester. Getting each week constructing feedbacks and working directions that greatly contributed in the achievement of this project. I would also like to thank Jean-Etienne Charbonnet that worked with me on the physics aspect of the project and with whom I could exchange ideas and solve problems.

8 Appendix

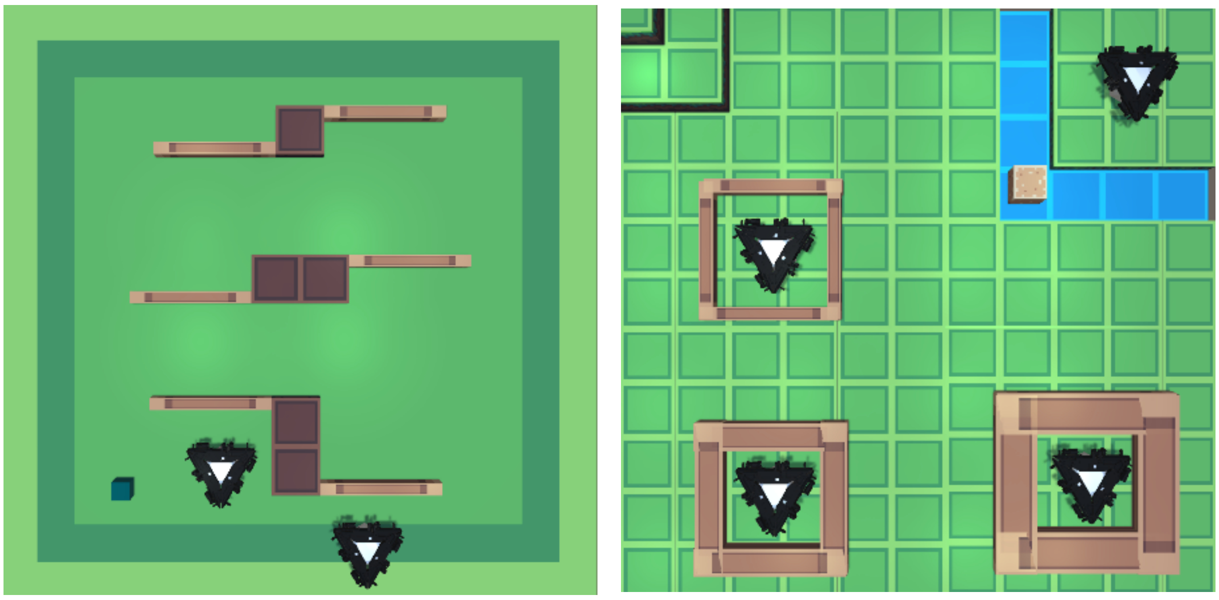


Figure 8: Obstacles examples

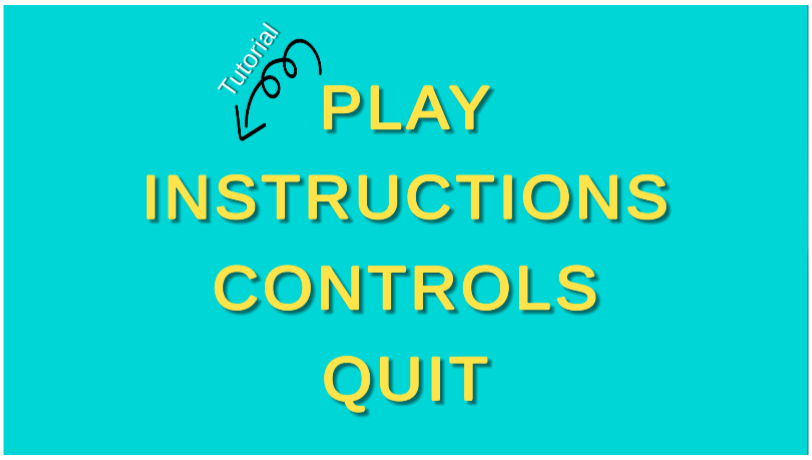
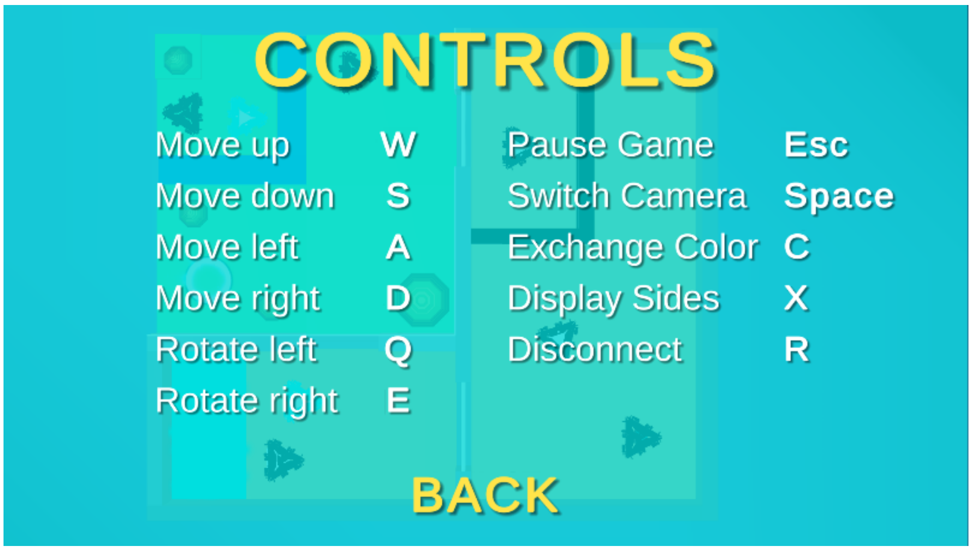


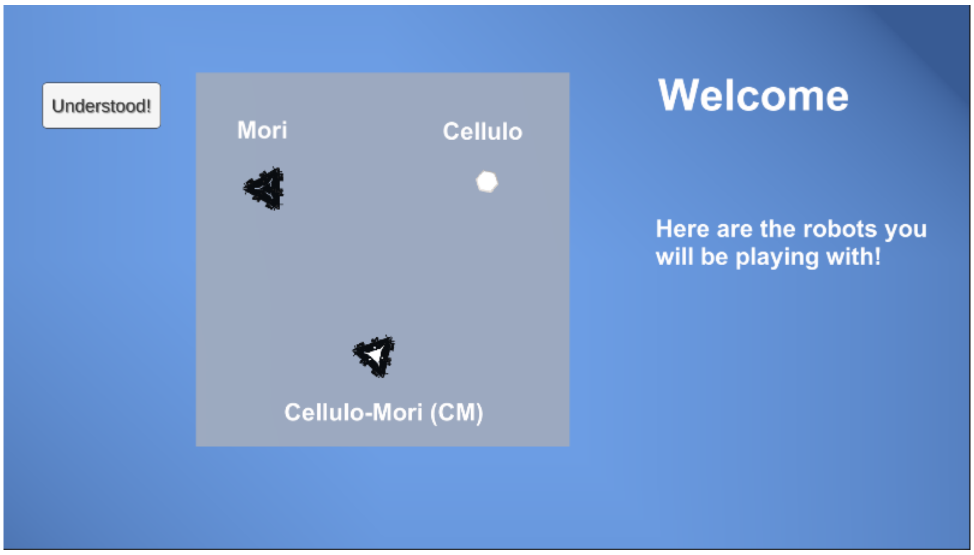
Figure 9: Main menu



CONTROLS

Move up	W	Pause Game	Esc
Move down	S	Switch Camera	Space
Move left	A	Exchange Color	C
Move right	D	Display Sides	X
Rotate left	Q	Disconnect	R
Rotate right	E		

Figure 10: Control menu while playing



Welcome

Here are the robots you will be playing with!

Figure 11: Tutorial