EPFL

CHILI
computer-human interaction
in learning and instruction

LEARN
Center for Learning Sciences

Unity-based Multi-User Game Implementation for Cellulo-Rehabilitation

Author:
Michael Roust

Supervisor(s):
Arzu Guneysu, Hala Khodr

Professor:
Pierre Dillenbourg

June 11, 2021

Contents

1 Introduction **1**

2 Game **1**
2.1 Players . . . 3
2.1.1 Local player . . . 4
2.1.2 Remote player . . . 4
2.2 Game Modes . . . 4
2.2.1 Competitive . . . 4
2.2.2 Cooperative (co-op) . . . 4
2.3 Collectibles . . . 5

3 Implementation **6**
3.1 Building upon the previous project . . . 6
3.1.1 Reducing Lag . . . 6
3.2 Cellulo Robot Mocking . . . 7
3.3 Game flow . . . 7
3.4 Movement and navigation . . . 8
3.4.1 Outer walls . . . 8
3.4.2 AI Chasing . . . 8
3.4.3 Streamlined map integration . . . 8

4 User interaction study **9**
4.1 Participant survey results . . . 9
4.2 Log data . . . 11
4.3 Observed player behaviours . . . 12

5 Future Works **12**

6 Conclusion **14**

Abstract

The projects aims to develop of a fully fledged online Pac-Man game with Cellulo robots. The game is intended to be used as a form of gamified rehabilitation and would allow patients to perform their rehabilitation while playing and interacting with friends or relatives over the internet. A pilot testing trial was performed with the end product of the project that proved that the game has a great potential in improving the rehabilitation experience of patients.

1 Introduction

The Cellulo robots are an excellent base for a variety of tangible interactive applications such as learning, rehabilitation or any other imaginable application. This project focuses on the rehabilitation part with a goal of creating an online two player Pac-Man game with Cellulo robots. With the goal of being a form of gamified multiplayer rehabilitation at a distance. Thus, allowing relatives and friends to connect and play with their loved ones while also helping them with their rehabilitation from anywhere in the world. This is especially relevant in view of the current social distancing measures which have especially isolated the elderly population, a majority of all rehabilitation patients.

The project builds upon a previous semester project [2] at CHILI lab which provided the basis for an online Unity game with Cellulo robots.

Throughout the project a large emphasis has been placed on simplicity in both code and game-play and making the game-play as natural as possible.

2 Game

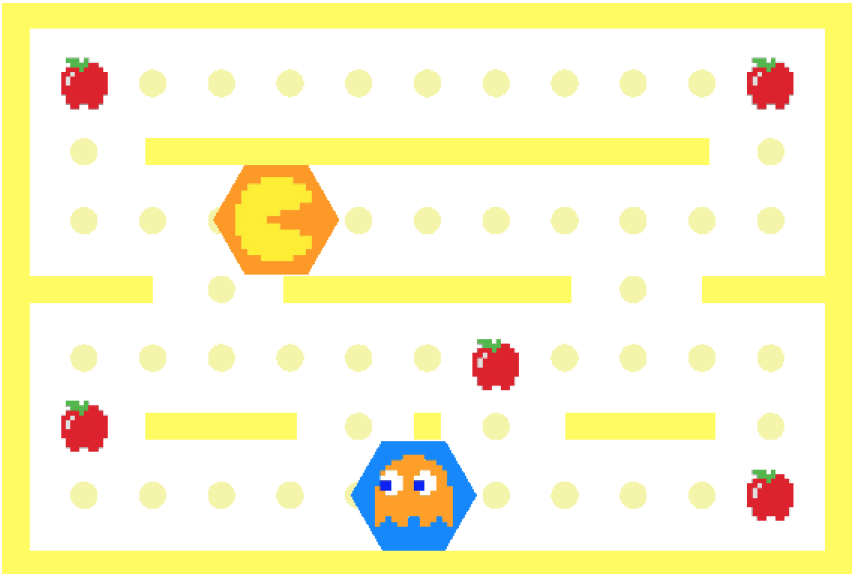

Figure 1: Game Screen in competitive mode on small map

The game consists of two games modes. A competitive game mode and a cooperative game mode. Which are further described later.

Both game modes require two players, one player has to physically move the tangible Cellulo robots on the printed map and the other player remotely controls a virtual robot using the arrow keys on his computer. This virtual robot motions are mapped to the motion of a real robot on the tangible physical side of the game. In both game modes there are two teams: the Pac-Man(s) and the Ghost. The goal of the Pac-Man(s) is to collect all the collectibles. The goal of the Ghost is to catch the Pac-Man(s). When a Ghost catches a Pac-Man the Pac-Man(s) lose all their collectibles and have to collect them all again, while the game timer isn't reset. The game is won when the Pac-Man(s) collect all the apples (game win screen 2). To not frustrate the patients there are no lose conditions or lives. Specifically, the use of lives is undesirable as the game is meant to be played by patients. However, to maintain an element of challenge the total time to win the game is displayed when the game is won, which is meant to challenge players to beat their record and replay the game to try to beat it faster and faster. The game timer starts when a Pac-Man collects an apple for the first time.

There is a tangible game map printed on paper where Cellulo robots can move on top and a representation of that map shown on the game screen, where the positions of Cellulo robots are synchronized with the game and shown on the game screen as hexagons with either a Pac-Man or Ghost symbol to indicate the corresponding team. LEDs on the Cellulo also correspond to the colors of the hexagons displayed on screen to easily identify which Cellulo is which.

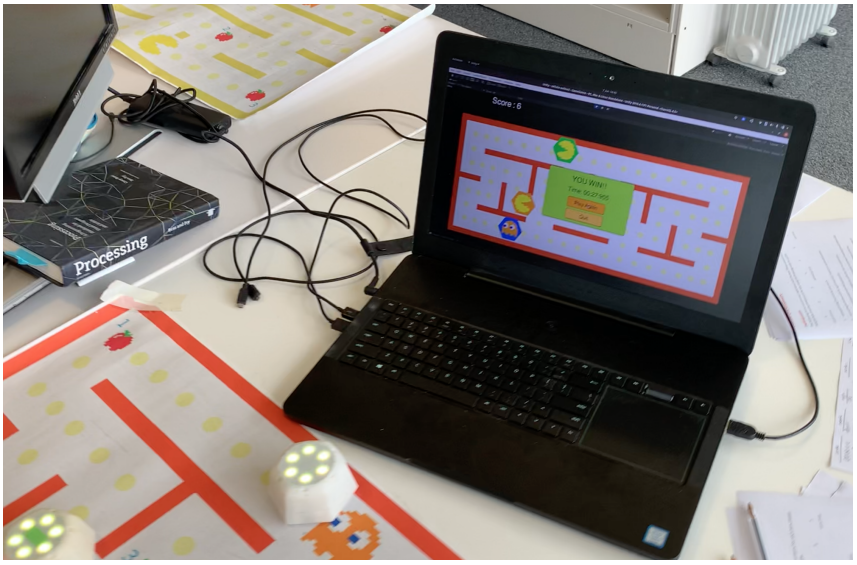

Figure 2: Win - robots flashing in yellow victory color

2.1 Players

Since the game is meant to promote social interaction as well, the game is intended to be played while a zoom video or voice call allowing participants to communicate. Either taunting each other in the competitive mode or discussing strategies in the cooperative mode.

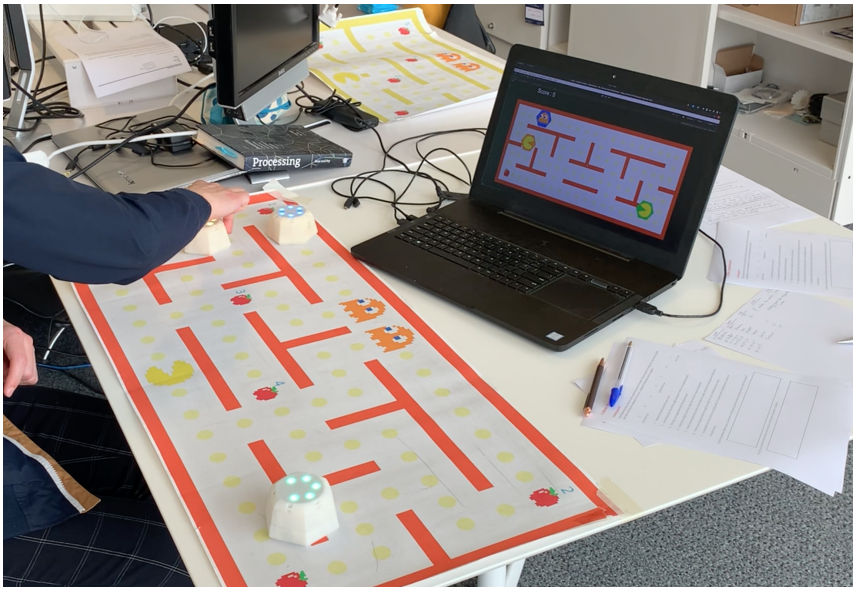

Figure 3: Tangible side where the local player controls a Pac-Man

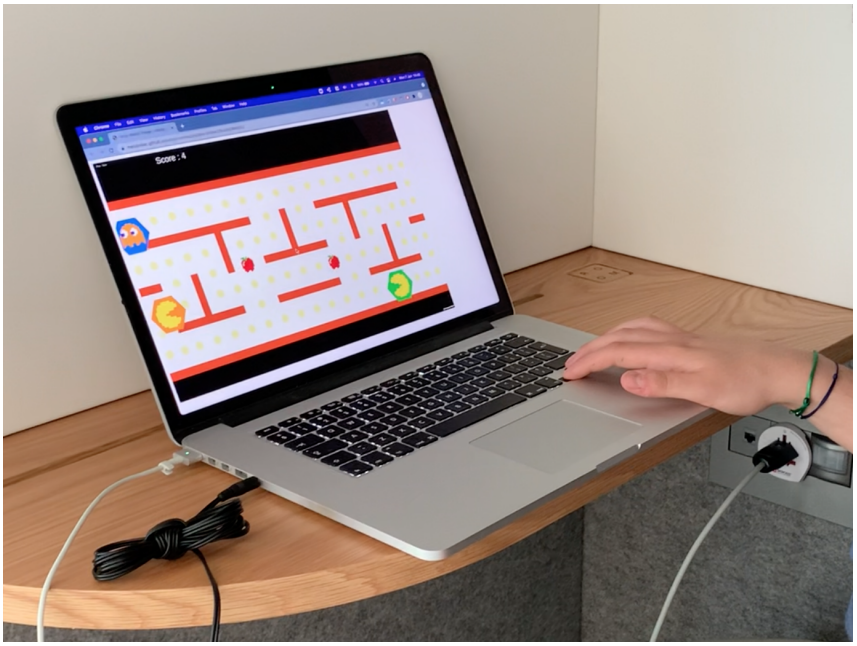

Figure 4: Online side where the remote player controls either a Ghost or Pac-Man depending on the game mode

2.1.1 Local player

Note: Throughout the report the term "Cellulo entity" will be used to reference the unified entity that is both the real tangible Cellulo and its representation on the game screen.

The local player (who's computer acts as the main controller for the game) plays the tangible side of the game with the Cellulo robots (see figure 3. The player controls the position of his Cellulo entity by physically moving the robot on the printed map.

2.1.2 Remote player

The remote player remote controls the Cellulo robots at a distance using his computer's keyboard and can see the game map and robots on his screen. The online player can be located anywhere in the world and connects to the local host computer using Photon PUN2 [4].

2.2 Game Modes

2.2.1 Competitive

The competitive game mode only requires 2 Cellulos. Here the local player controls the Pac-Man and the remote player controls the Ghost (see figure 6). As usual the Ghost tries to stop the Pac-Man from collecting all the apples for as long as possible.

2.2.2 Cooperative (co-op)

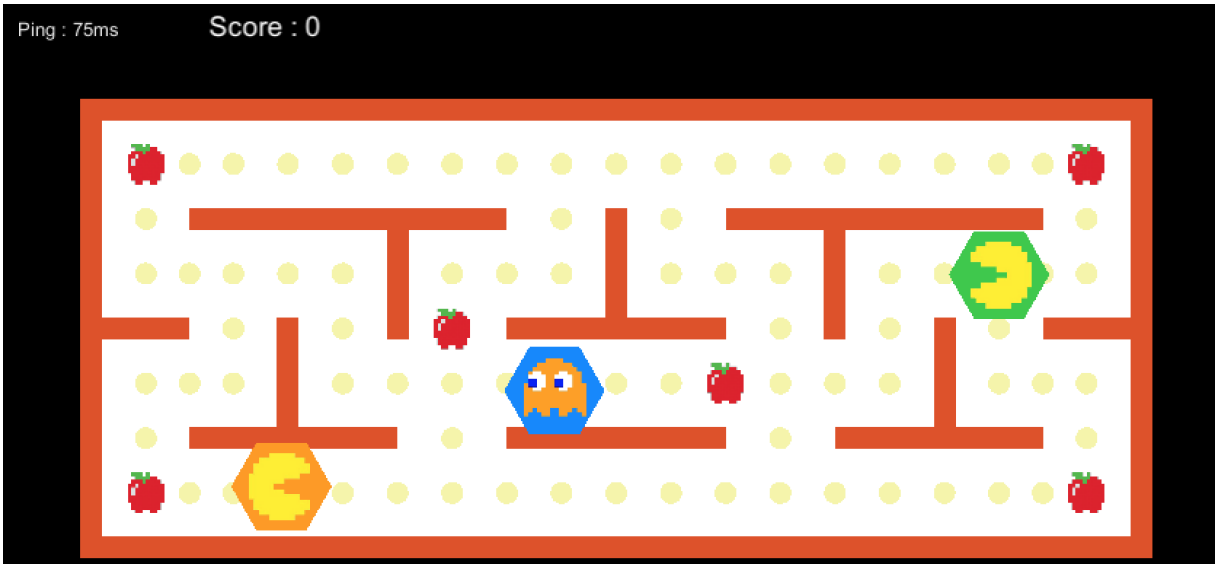

Figure 5: Game Screen in co-op mode on large map

The cooperative game mode requires 3 cellulos. Here both the local and remote players control 1 Pac-Man each and this time the Ghost is an AI that chases the closest player.

2.3 Collectibles

Since, the tangible game map is static and cannot have disappearing collectibles it was needed to design a clever way to represent which and how many collectibles have been picked up without requiring the local player to constantly check the computer screen which would create a split attention effect which goes against sensible game design principles [1].

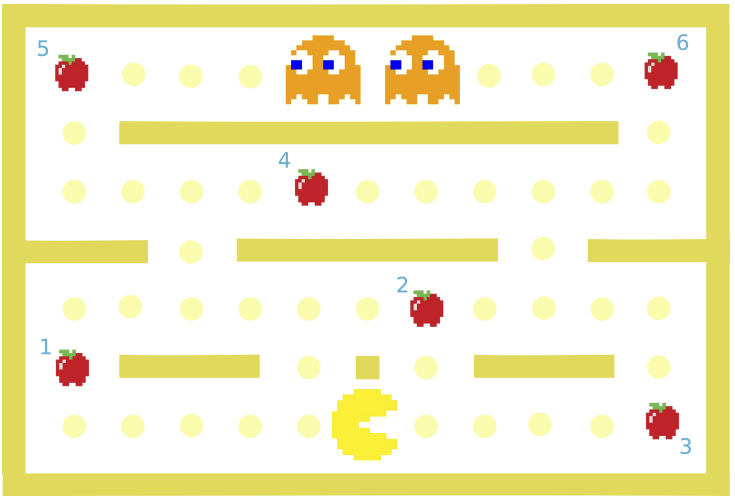

Figure 6: Small game map with numbered apple collectibles (printed tangible map)

The solution was to use the 6 LEDs on top of the Cellulo robots to indicate which apples have been picked up. Every time an apple the LED corresponding to the same index as the number besides the apple (shown in figure 6) on the tangible map is lit up. Adding numbered stickers to the Cellulo LEDs will allow players to make the association. This is unlike in a regular Pac-Man game where both the dots and fruits are all collectibles with different score values and even effect, but appears to be a justified sacrifice to adapt the game to our constraints.

The numbers aren't shown on the game screen itself as there the apples disappear and the extra information isn't required.

3 Implementation

Since the physics of the Cellulo entities are defined by the real Cellulo robots there is no usage of Unity's Physics system, except for some cases with collision detection and mocking the real Cellulo robots. Thus, in the communication between host and remote computers only position data is sent which is always taken from the real positions of Cellulo robots.

Also, through out the project an extensive usage of C# Serializable fields allows rapid tweaking of any relevant parameters in the game.

3.1 Building upon the previous project

Despite having basic knowledge of Unity it was quite challenging to get started with the project due to having to understand 3 different components simultaneously them being: the Unity game itself, Photon Unity Network 2 (PUN2 in short, the online communication platform) and the Cellulo library.

The initial codebase was largely redesigned, optimized and further modularized with the goal of allowing future work on the project to be more easily carried out. The UI was also overhauled an simplified as some previously used game modes were no longer needed.

I did find that Unity sometimes hinders the application of clean programming principles. It wasn't possible to completely separate and modularize all the 3 components of the game.

3.1.1 Reducing Lag

In the base project a latency of about 1-2 seconds was observed between a key press on the keyboard and the response of the cellulo on both the local and remote side of the game. The issue was caused by overloading the bluetooth connection to the Cellulo by sending sending new inputs at a too high frequency. Two solutions were explored:

- Only send a new input when it differs significantly from previous input.
- Simply use a counter and conditional statement to only send inputs once every few game updates. (In Unity a game update occurs at every frame, usually about 60 times per second.)

Even though, the 1st option seems more advanced it significantly increased code complexity as logic determining if a new input was to be sent had to be carefully designed to avoid problems such as imparting a residual velocity when all controls were released. Therefore, the later option was chosen in the aim of reducing code complexity and simplifying debugging. With this fix lag dropped to about 200ms except for the remote player who's robot's position appears to be smoothed out by Photon library which appears to seriously increase the perceived lag. Also, when playing the game through WebGL there appears to contribute slightly to the lag due to extra network layers in the messaging protocol increasing message transmission time.

3.2 Cellulo Robot Mocking

The `CelluloEntity` class serves as a single module through which all interactions with the Cellulo entities are done: moving the Cellulo, changing its lighting, ordering it to follow a given path (a list of coordinates). The module also contains no code related to online communication and could be easily used in any game where we want to synchronize the real Cellulo position with a Unity GameObject on screen.

In order to speed up the development process the decision was made to implement this class such that a `isCelluloVirtual` parameter can be passed to the class upon initialization. If true the class will not attempt to connect to any Cellulo robots and instead simulate all Cellulo movements using Unity's Rigidbody2D (an exception to not using Unity's Physics). Therefore, allowing to develop, test and play (a non-tangible) version of the game without needing any Cellulo robots. The entire AI navigation was implemented and debugged using this and worked right out of the box when using it with real Cellulos. The addition can be of great value for any further developments of Unity games with Cellulo robots.

3.3 Game flow

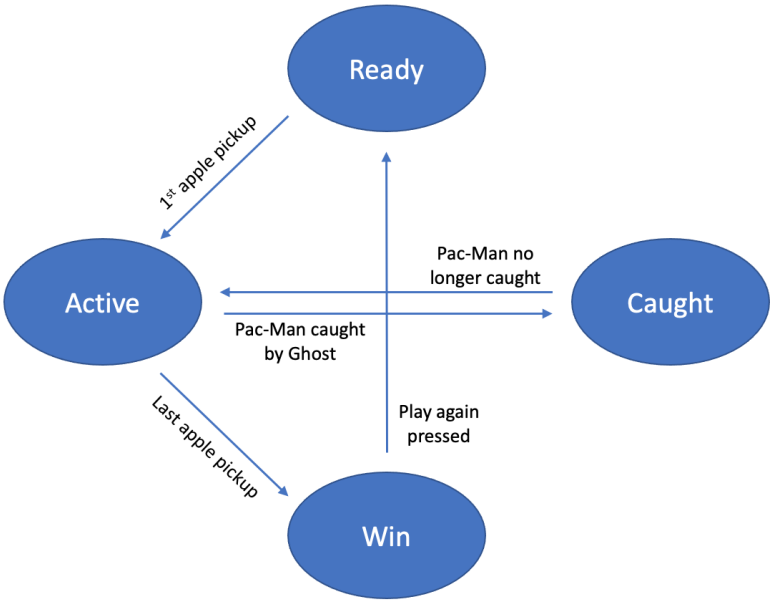

Figure 7: Game states

The behaviours and flow of the game are dictated by the finite automata illustrated above (figure 7). The game timer starts when the very first apple is picked up transitioning the game from ready to active. The collection of apples and the ability to get caught by the Pac-Man is only possible during the active game state.

3.4 Movement and navigation

3.4.1 Outer walls

Since the physics of robots aren't dictated by Unity's physics the behaviour of walls has to be manually defined. Especially, in order to stop Cellulo robots from running off the game map the outer walls of the map restrict movement in a specified direction if a Cellulo entity is in contact with a wall.

3.4.2 AI Chasing

Autonomous AI chasing of the closest Pac-Man was implemented using Dijkstra's shortest path algorithm. A C# implementation of the algorithm [3] was adapted to be used by the game. A navigation graph is defined through Unity (see 3.4.3 which is then translated into a graph stored as a list of vertices and edges and used by the algorithm. The algorithm then outputs a list of 2D vectors encoding the path to be taken which is then send to the `SetGoalPath` method of the AI's `CelluloEntity`.

The code also contains an implementation of A* search but its usage was deemed unnecessary as even in the large map there are only 84 nodes and paths are only periodically recalculated.

3.4.3 Streamlined map integration

The scale of the game map is also identical to that of the Cellulo's library (millimeters), removing the need for any rescaling. It should be noted, that Unity's coordinate system has its y coordinates inverted compared to that of the Cellulo library.

The parameters of elements used for navigation are all fully specified through the Unity editor and thus there is no need for any hardcoded values inside the actual code and logic of the game. This is achieved through the usage of Serializable fields and Game Objects.

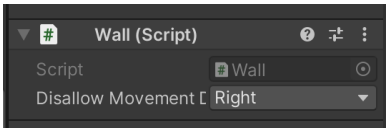

Figure 8: Wall configuration

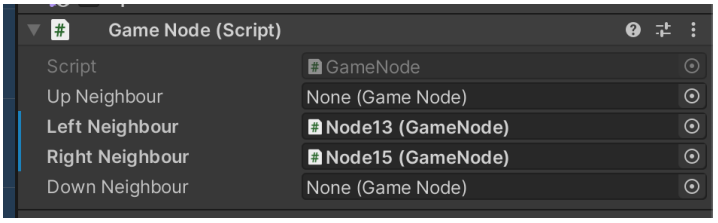

Figure 9: Game node configuration

More specifically, the outer walls have an enum field that can be set to determine towards which direction do they disable movement. In figure 8 is the configuration of the right outer wall of the game map, which specifies that if in contact with that wall any further movement towards the right is to be prevented.

For the Game nodes (which are all the yellow dots on each map) used for the navigation. They are configured by selecting their neighbors though Unity. The game node configuration data, corresponds to a vertex and its adjacency list.

All of these design choices allow to quickly edit or integrate new maps into the game without even having to modify any code.

4 User interaction study

A pilot experiment was held with volunteers to gather data and feedback on the game. A total of 10 volunteers participated each playing 2-4 games of both game modes on both the tangible side and online side of the game. Participants were all healthy individuals with ages ranging from 20 to 40, with full unimpaired motor function. The players were placed in separate rooms and could only communicate through voice chat to simulate the conditions that the game is intended to be played in. Each game's duration and number of times caught during a game was logged for each game and participants were given a survey about their thoughts on the game at the end of the experiment. The players only played on the large game map as it seemed to be the most fun to play on and to reduce the different number of configurations we would test. Since, we already had to have both participants try both game modes and both tangible and online play, which amounted to 4 different configurations for each pair of participants.

Overall, excellent feedback was received. The application functioned very well and participants had good fun playing around with this experimental project. Most suggested improvements were features that couldn't be implemented in the given time frame.

4.1 Participant survey results

Very positive feedback was received and participants found the game fun, well integrated and intuitive (see table 1). They also all agreed that the game would fulfill its designed purpose of social interactive gamified rehabilitation very well.

Table 1: Qualitative survey results

	Average response on a (0-7 scale). 0 being not at all true and 7 being very true
I enjoyed doing the activity very much	6.9
I am willing to do this activity again because I think it is somewhat useful	6.1
I felt that we cooperated with the second user	6.4
I think that doing this activity might be useful for social connection	6.6
I would recommend this activity to the elderly to play with their friends/grandchildren etc.	6.7
I think that doing this activity might be useful for arm rehabilitation at home	6.7
It is possible that this activity could be useful to improve the rehabilitation process of friends/family	6.4

Participants were also asked to express their thoughts on the interaction with the game elements and players in writing. A summary of their responses is shown in tables 2, 3. Apart from the positive comments participants did mention room for improvement. It was observed that lag was still too high on the remote client which will be something to improve in future versions. It should be noted that during the experiment the remote user was using the WebGL client which as mentioned adds some latency in the communication. Without any changes to the code a using a standalone client compiled for the local operating system would already improve latency. Furthermore, several players commented that inner walls of the map should serve more that just a visual purpose and penalties for traversing them should be added. The comments also highlight that the Ghost was often too slow or in one situation too slow (further described in section 4.2). This issue is probably in fact not caused by the robot being too slow but actually because the player moving the Cellulo by hand can move it at significantly higher speeds than the robot can move at itself. This problem should be resolved by applying some speed limit that would penalize users who can use their perfect motor function to move robots faster than intended and cheat.

Table 2: Summary of comments on game, game modes and interaction with robots

Number of times the comment was made	Comment
4	Intuitive and user friendly interface
4	High lag on remote client
3	Great approach with the two different game modes
3	Interaction with robot felt natural
3	Competitive mode was the most fun
3	Should add rules for penalizing trespassing walls
2	Ghost speed should be adjustable or dynamic
2	Integration of physical map and game was seamless
1	The resistance of the robot explains its potential usage for rehabilitation

Table 3: Summary of comments on interaction between players

Number of times the comment was made	Comment
4	Great fun laughing and playing with friends
2	Co-op mode encouraged communication and group co-ordination
1	Interaction though both the game and voice call bring players together and enhance both game modes
1	Enjoyed the nostalgic feeling of playing Pac-Mac
1	Adding a video stream and displaying player names would improve social connection

4.2 Log data

Throughout the experiments a total of 41 games were played. 20 Games in adversarial game mode and 21 games in cooperative mode. On average competitive games lasted 47.2 seconds (standard deviation 29.1s) and Co-Op games lasted 40.3 seconds (standard deviation 31.9s) and each participant got to play 2 games in each configuration.

In the competitive mode players controlling the Pac-Man got caught 0.36 times per game on average. Whereas, this value is 1.61 for the Co-Op mode. This can be explained by the fact that remote players had trouble running away from the AI Ghost despite its movement speed having been set to be 10% slower than that of the remote player. On the other hand players controlling the Pac-Man by hand could quite easily outrun the AI Ghost. Imposing speed limits and some tweaking of game parameters could improve this in the future. Another explanation for getting caught by Ghost more often is there are now 2 targets the ghost may catch. Nevertheless, cooperative games were probably shorter than adversarial games since two Pac-Mans can collect the apples faster than one.

4.3 Observed player behaviours

Several strategies and behaviours were discovered or adopted by participants when playing to improve their performance. Some pairs played the game and tried not to cheat much whereas a few other players used all exploits available. Moving the robot by hand at much higher speeds than other robots can reach made games pretty easy for the local player. In one instance the local player lifted the robot above a ghost and placed it on the other side of the map, resulting in an apparent teleportation of the robot on the game screen.

Another interesting strategy was to defend an apple by blocking the Pac-Man's path. It went as far as even standing still on top of an apple, completely preventing the Pac-Man from picking it up. Surprisingly, no players had the idea to cheat by moving another Cellulo robot than the one assigned to them. This could have been a funny counter to the previous exploit.

5 Future Works

There is a vast number of possible improvements and features that could be added to the game. Significantly, more than could ever be attempted in the scope of this project. Hopefully, these may serve as inspiration for future works.

- **Dynamically resize map size to always fit inside of game screen**: Currently the map doesn't scale properly and in some unusual cases won't fit inside the window. Fixing this would allow placing a video stream of the players side by side with the game.
- **Adding more maps**: So far the game has the small and large Pac-Man map. There already exists a medium map of the game which could be added or even designing a whole new ever greater map and integrating it in the game.
- **Further reducing communication latency**: As noticed by participants in the pilot experiment there was a very noticable latency between the movement of the Cellulo robot and its position being updated on the WebGL application of the remote player. Apart from just using a standalone application to avoid the additional latency due more network layers in the communication it may also be used due to an attempt by Photon to smooth out the movement of the Cellulo by using interpolation.
- **Cheating detection and penalties**: Again as noted in the experiment feedbacks the current implementation of the game does nothing penalize players who are cheating. Implementing code for detecting and penalising the traversal of walls, lifting the cellulo or having the local player drag the cellulo over that map at a much higher speed then the remote cellulo can possibly reach. For example: a good penalty for such transgressions would be loosing an apple.

- **On rails navigation**: It may be worth exploring the possibility to impose on rails movement similar to the path navigation of the AI Ghost as it would remove the possibility for the remote player to cheat. However, it may not be desired as the local player on the tangible side will always be able to cheat since he can drag the robot over the walls drawn on the map.
- **Real physical map walls**: It may even be possible to add cardboard walls to the game map. This would overall fix most problems related to cheating. The only possible exploit left would be to lift robots over walls but that should be simple to detect. Players will no longer have the option to cheat (and potentially lose one apple) in order to avoid getting caught and lose all their apples. Which forces the player to make tricky strategic decisions and increases fun factor.
- **Better logging**: Logging more information such as player positions and etc.
- **Further code cleanup and modularization**
- **Integrating the new Multiplatform Unity Cellulo library**: Allowing the game to be hosted on operating systems other than Linux.
- **Integrating more Cellulo features** such as haptic feedback and displaying battery levels on screen.
- **Game Scoreboard**
- **More ghost behaviours**: Currently the ghost always chases the closest player to itself. The game could be improved by adding some randomness in the robots decision making or implementing different chasing behavious such as running away from Pac-Man's, random pathing and etc. Much like in the original Pac-Man game.
- **Multiple AI ghosts**: The current implementation of AI pathing is limited as it cannot avoid robots on a path it may want to take. This means that if we were to add 2 AI ghosts to the game they may collide with each other (depending on behaviour). For example: this could done by recalculating paths while removing the edges other robots are currently on during the pathing calculation.
- **More game modes**: Extending the code to have variable amounts of players and more game modes. Here are a few examples:
  - **"Coin"/Apple rush**: This would be a competitive game mode with or without a ghost where players would compete to grab the most apples in a limited amount of time. Apples could appear randomly on the map at random intervals.

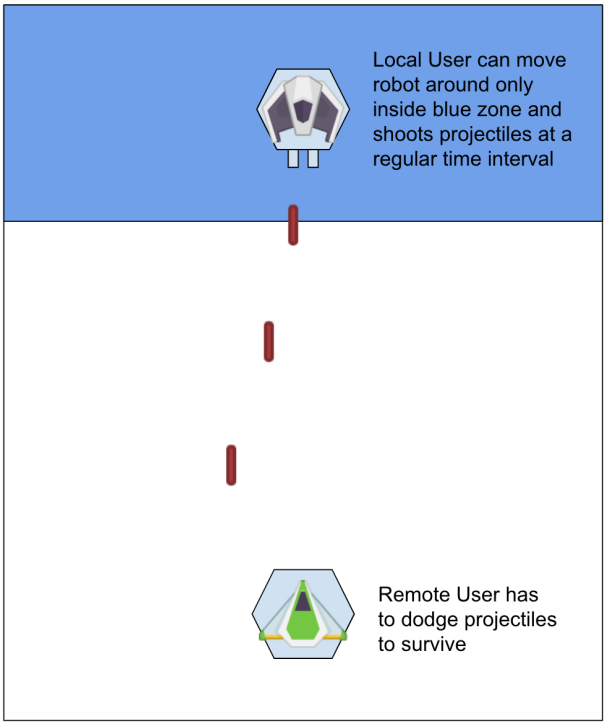

Figure 10: Space invaders

– **Space invaders**: During the project a completely different game mode was imagined. The idea would be to implement a version of the classical space invaders arcade game. However, since the game would have moving projectiles it would not be possible to display them on the static map and thus it would be challenging to display all the needed information on the game map and avoid having the player divert part of his attention to the game screen to keep track of the projectiles. This limiting factor could be partially solved by always having the remote player be the one dodging projectiles which he will be able to see on screen. Otherwise, it may also be possible to use a projector to project dynamic components onto the game map, such as the projectiles. In fact this addition could offer vast new possibilities for game modes that were previously irrelevant due them being too reliant on dynamic components in addition to being applicable for this suggested game mode.

6 Conclusion

Overall, the project was a success. A full functional game has been developed and a successful pilot test conducted. Participants in the testing found that the game could indeed be used for rehabilitation purposes. Latency issues have been reduced to reasonable level and with some more work the issue could potentially be completely solved. Excellent feedback was received and with some extra features and improvements in the UI the

game could be deployed in larger trials. The project has been further modularized and should be quite intuitive to build upon. There is a vast number of possibilities for further improvement and expansion of the project.

References

[1] Arzu Guneysu Ozgur et al. *"Gamified Motor Training With Tangible Robots in Older Adults: A Feasibility Study and Comparison With the Young". In: Frontiers In Aging Neuroscience 12*. This article is licensed under a Creative Commons Attribution 4.0 International License, p. 59. 2020. DOI: 10.3389/fnagi.2020.00059. URL: http://infoscience.epfl.ch/record/277462.

[2] Mehdi Akeddar. *Development of an Online Multiplayer Gaming Platform for Cellulo*. 2021.

[3] Kristian Ekman. *Pathfinding Algorithms in C#*. URL: www.codeproject.com/Articles/1221034/Pathfinding-Algorithms-in-Csharp.

[4] *Photon PUN 2*. URL: www.photonengine.com/pun.