Promoting IMGD Through Play

Sean Briggs and Bailey Sostek

Brian Moriarty, Advisor

Abstract

This paper covers our research into creating an exhibit to showcase the Interactive Media and Game Development program at WPI. After considering a variety of technologies, we settled on using HoloPlayer One, a new autostereoscopic device that projects interactive 3D images into the space between the screen and viewer. Because of its wide range of software support, this device has the potential to provide an attractive, accessible way for IMGD students to showcase their work.

Acknowledgements

This paper would not have been possible without the help of Brian Moriarty, to whom we are extremely thankful. His knowledge, enthusiasm and industry experience provided key insights into 3D display technologies as well as the structure of this paper.

Additionally, we would like to thank Joe Rose for encouraging Bailey Sostek through high school to peruse greatness and not let the size of a paper limit what is possible. Finally, we would like to thank Bill Chamberlain for introducing Bailey to computer science, and encouraging him to pursue it as a career.

Contents

1. Introduction

WPI is an environment filled with thousands of students creating projects of which they are justly proud. Many of these projects could benefit from feedback provided through greater exposure to the WPI community and the greater Worcester area.

In WPI’s Interactive Media and Game Development program, students maintain portfolios throughout their academic careers to showcase the projects they have created. Several classes, such as IMGD 2900 (Digital Game Design I) and 3900 (Digital Game Design II) facilitate a development context which encourages students to create elegant and concise games in a game engine called Perlenspiel. Some of these games feature excellent design which could potentially serve as exemplars of the project work undertaken by the IMGD program.

The WPI community would benefit from an interactive display device featuring student-made games. Such a device would promote the IMGD major and provide a means for students to publicly present their work and receive valuable feedback.

Students and faculty would not be the only demographic to benefit from such a device. We believe that an attractive display set up in a public space which elicits play would also delight campus visitors, particularly prospective students. Enticing visitors to interact with student game projects would leave a positive impression of WPI in general and the IMGD program in particular.

Such a display might take the form of an arcade cabinet containing a device for interpreting and playing Perlenspiel games. The choice to use an arcade cabinet is deliberate, because their immediately recognizable appearance signifies an opportunity to play. Norman points out that certain objects elicit action through their design, noting that “Affordances determine what actions are possible. Signifiers communicate where the action should take place.” (Norman 14) This suggests that when someone sees a device resembling an arcade cabinet, they are likely to assume that the device can be used to play games. An IMGD-branded arcade cabinet would establish and/or reinforce an association of IMGD with games and play.

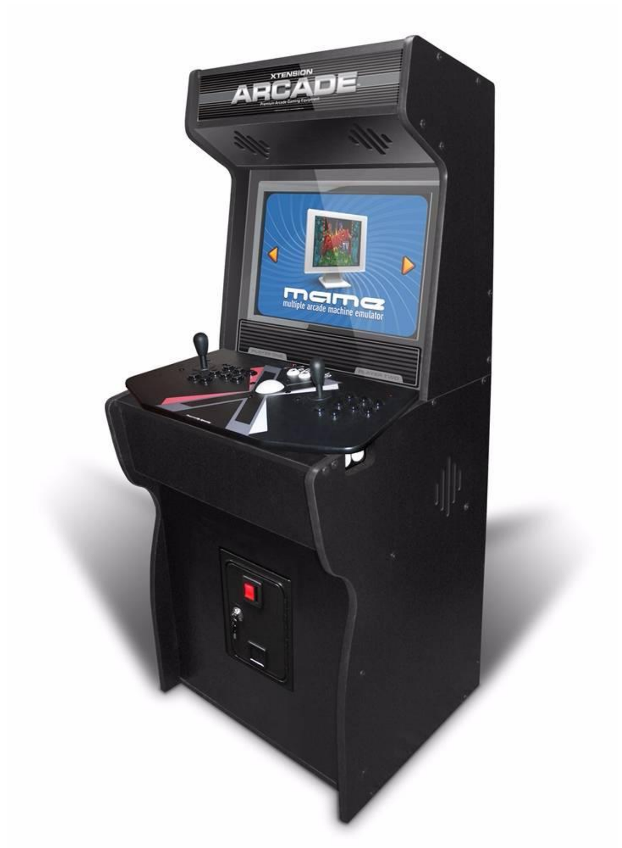


Figure 1. The above figure represents what the final arcade cabinet may look like. Source: URL.

Figure 1 depicts a device similar to that which we imagine constructing. The cabinet would be free-standing, and require only a standard AC power outlet. Internet communication will be facilitated via WiFi, and automatically initialized when the cabinet is turned on.

The device itself would be developed by students collaborating in the context of a Major Qualifying Project (MQP). Specific aspects of the project, such as art for the side panels as well as games to run on the cabinet, could be completed by students enrolled in IMGD design courses, or optionally through Independent Study Projects (ISPs).

Every aspect of the display will be designed and assembled by WPI students. The cabinet itself would be manufactured by mechanical engineering students. The art on the sides of the cabinet will be executed by IMGD artists. The games offered for play will be produced by IMGD designers, and engineered by IMGD Technology students.

The device is expected to undergo heavy usage, and will be constructed accordingly. Similar to interactive museum exhibits, all touchable and moving components will be well-protected and easily replaceable to ensure that the device will remain functional for years to come.

We believe such a display has the potential to become a signature product of the IMGD program. Wherever it is exhibited, it will showcase what WPI's project-based learning is capable of accomplishing.

1.1. Primary display option

A wide range of display technologies have been investigated to determine which would work best for this application. These included LCD panels, NeoPixel LEDs, ferrofluid, touch interfaces, various projection configurations, and holograms. One of the most intriguing options we discovered is the HoloPlayer One, a device recently introduced by Looking Glass Factory.

The HoloPlayer One combines a digital lenticular display with a retroreflector to create full-color 3-dimensional entities that appear to "float" in the space in front of the screen. The device is *autostereoscopic*, meaning that no special glasses or viewing aids are required to perceive the 3D effect. It is also interactive; integrated motion sensors allow users to "play" with the virtual objects it produces using simple hand gestures.

The magical imagery produced by the HoloPlayer is quite striking; few devices similar to it have been widely deployed. Passersby are likely to be intrigued by its novel display, and be drawn to interact with it.

2. Design considerations

The HoloPlayer One has relatively low resolution compared to conventional 2D displays. The LCD monitor that drives it features a high-definition array of 2560 x 1600 pixels, but the horizontal view is partitioned into 32 sub-zones in order to create the 3-dimensional effect. This yields an effective image resolution of only 267 x 400 pixels. (Higgins) Projects chosen for display on the device must therefore be designed to accommodate this extremely limited level of detail.

2.1 Perlenspiel

Perlenspiel is a Web-based game engine which limits users to a raster of 32 x 32 jumbo-sized pixels. These pixels are referred to as “beads,” a nod to the engine’s original inspiration, Herman Hesse’s novel *Das Glasperlenspiel* (The Glass Bead Game, 1943). (Moriarty)

The engine was developed in 2009 by IMGD Professor of Practice Brian Moriarty for use in his digital game design courses at WPI. It was introduced to the academic community in a keynote lecture delivered at the Education Summit of the 2012 Game Developers Conference in San Francisco. Over the years, students at WPI and several other institutions have used Perlenspiel to develop thousands of simple, elegant games such as *W(h)ither* (Melville/Pederson, 2015).

Designers use a set of Application Programming Interface (API) commands to respond to user activity such as mouse movement and keypresses, change the color and other attributes of individual beads, and play sounds. The limited visual capabilities provided by the engine force students to focus on design rather than asset production, and make the most out of every game element.

Commercial game engines such as Unreal or Unity present developers with a myriad of possible projects, varying widely in scope, nearly all requiring significant audiovisual assets, design and code. Such engines are designed for teams of engineers, designers, artists and writers working together. A single person developing a full game with one of these engines would require expertise in multiple disciplines and a significant time investment.

Perlenspiel’s extreme constraints make the game production process considerably more manageable. Users do not need to worry about graphics, 3D modeling or library management. Their entire focus can be devoted to design. Using a grid of colored beads to convey the mechanics and meaning of a game becomes the primary challenge.

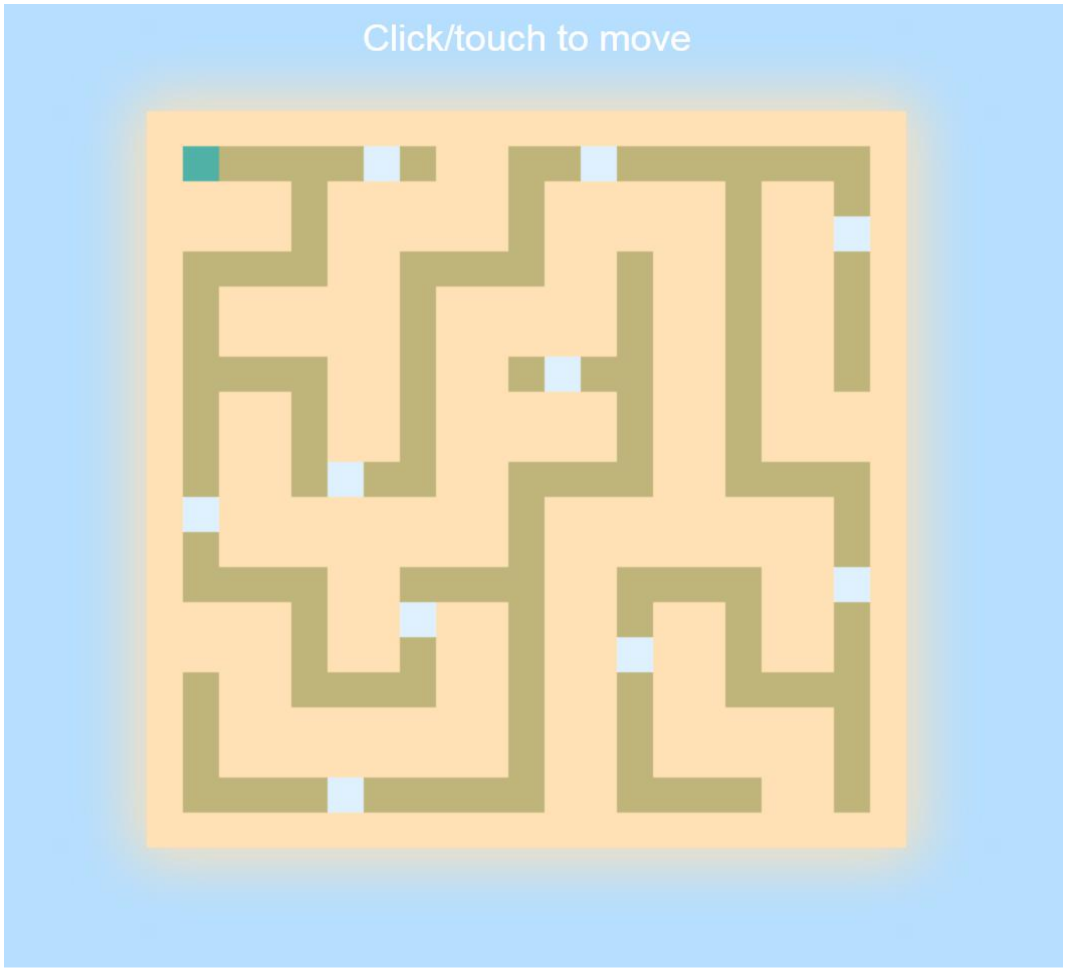
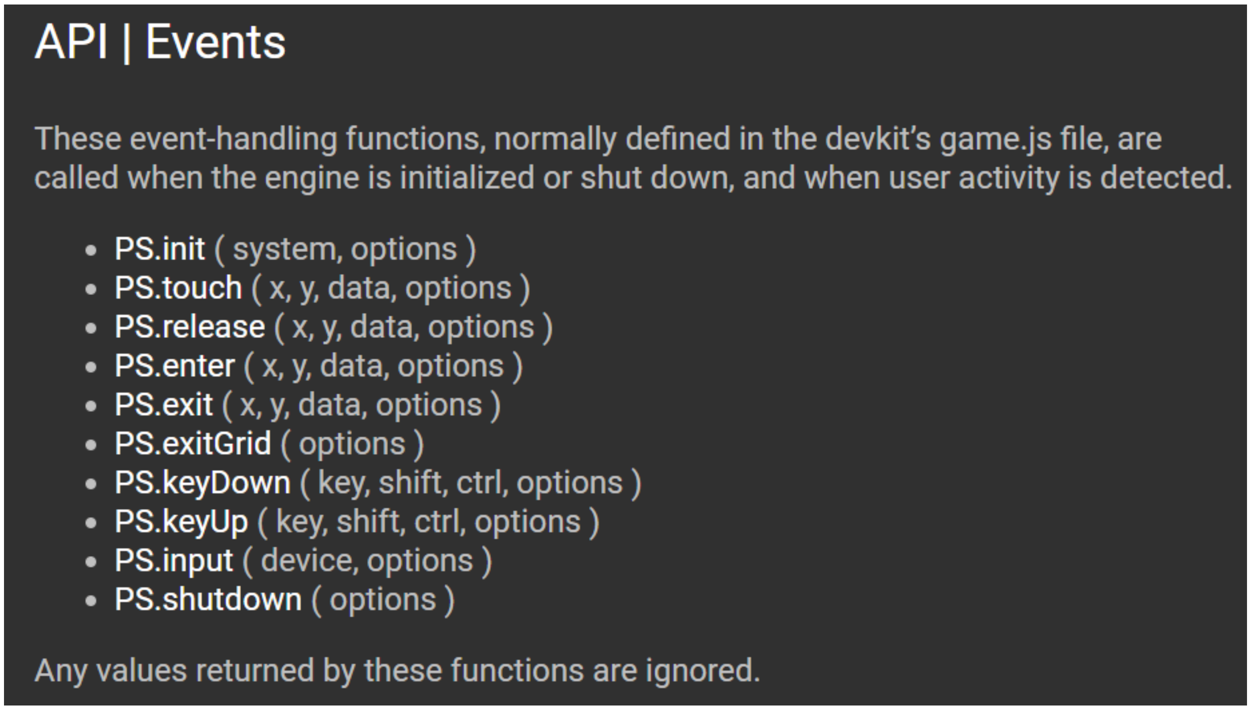


Figure 2. The opening screen of Gulls *(Sostek 2016).* Source: URL.

The 16 x 16 bead game shown in Figure 2 demonstrates how beads can be employed in a variety of creative ways. The light blue bead in the top left represents a player. The off-white beads represent seagulls. The tan beads represent sand. The addition of custom sound effects (crashing surf, gull cries) reinforce the evocation of an entire beach scene with only 256 colored squares.


API | Events

These event-handling functions, normally defined in the devkit's game.js file, are called when the engine is initialized or shut down, and when user activity is detected.

- PS.init (system, options)
- PS.touch (x, y, data, options)
- PS.release (x, y, data, options)
- PS.enter (x, y, data, options)
- PS.exit (x, y, data, options)
- PS.exitGrid (options)
- PS.keyDown (key, shift, ctrl, options)
- PS.keyUp (key, shift, ctrl, options)
- PS.input (device, options)
- PS.shutdown (options)

Any values returned by these functions are ignored.

Figure 3. A List of all Events that can happen in Perlenspiel 3. Source: URL.

It does not take much to make a Perlenspiel game. A single JavaScript file which implements one or more of the engine's ten event handlers (listed in Figure 3) is enough to produce a working application.

WPI is not the only school using Perlenspiel. One institution which has incorporated Perlenspiel into their undergraduate program is Full Sail University. Along with their game design offerings, they offer a *Programing Foundations* course that the engine as a tool for exploring the fundamentals of programming. The course's creator, Mark Diehr, has also produced a video series explaining programming by example with Perlenspiel. (Diehr)

Perlenspiel's grid of beads is easily accommodated by the limited resolution of the HoloPlayer. Almost any game created in Perlenspiel could be displayed. Incidental features of the engine could be implemented through other display technologies. For example, Perlenspiel allows users to specify a color for the background area surrounding a game. This could be accomplished with an array of RGB LEDs surrounding the HoloPlayer One's display area. Additionally, Perlenspiel allows users to display a single line of text above the grid. This feature could be implemented with an auxiliary LED panel.

2.2 Independent Study Projects

WPI allows for students to complete out-of-course activities under the supervision of a professor for credit. These projects are called Independent Study Projects (ISPs), and are usually completed over the course of a single term. Due to the limited amount of time in a term (seven weeks), developing a complete game in a traditional game engine is a difficult task to achieve. If a student was developing in Perlenspiel for an ISP, they would easily be able to design, develop, test, polish and deploy a full game by the end of the term. This allows for students with interesting game ideas to rapidly prototype and complete a games in Perlenspeil, then showcase it on the arcade cabinet to get user feedback.

An additional ISP required for this project would be the design of console art. As can be seen in Figure 1, the arcade cabinet will have large black panels on its sides. These side panels could be textured by an IMGD art student. The art would need to stretch around the whole console, featuring a WPI-branded illustration which is attractive and inviting.

2.3. Game Audio I

In IMGD 2030 (Game Audio I), students learn the basic tools for creating digital audio tracks for games. This course focuses on using the application Reaper to create individual sounds, and for mixing audio tracks for gameplay and cut scenes. The final project of this class is to mix an entire cut scene, which is then rendered as an MP4 video.

Student-made cut scenes could be rendered on the display and exhibited along with Perlenspiel games and other IMGD-produced content. Any independent audio tracks that were produced by students would also be able to be exhibited on the display through the device's integrated speakers. Opportunities for collaboration between classes are also possible. For example, a student in Digital Game Design I could create a visualizer for music produced by students in Game Audio I.

2.4. 3D Modeling I

In the course 3D Animation I, students create models and scenes with various applications, such as Autodesk Maya and ZBrush. This course teaches students the fundamentals of how to create and animate 3D models and scenes. The models produced in this class could easily be exhibited on the HoloPlayer as 3D objects suspended in midair, allowing users to interactively rotate them for inspection from any angle. This would provide an opportunity for students to showcase digital artwork that would otherwise be hard to exhibit.

2.5 Digital Game Design

In Digital Game Design I, students begin by making a toy, followed by a puzzle, and concluding with a game that must adhere to a series of challenging constraints. Throughout the class, students are forced more and more to think outside of the box while learning the capabilities of Perlenspiel. Almost any of these assignments could be featured on the arcade cabinet.

In Digital Game Design II, students make games incorporating telemetry for statistical analysis, culminating in a project that illustrates a classic fable, legend or aphorism. In this course, teams of students could compete and vote on which game they believed was most innovative, and have that game featured on the arcade cabinet.

3. Additional display options

3.1. LED matrices

LED matrices are a display technology which could easily be implemented into an arcade cabinet. Large arrays of individually addressable RGB LED lights called, “NeoPixels”(Adafruit) can be strung together to form a display surface.

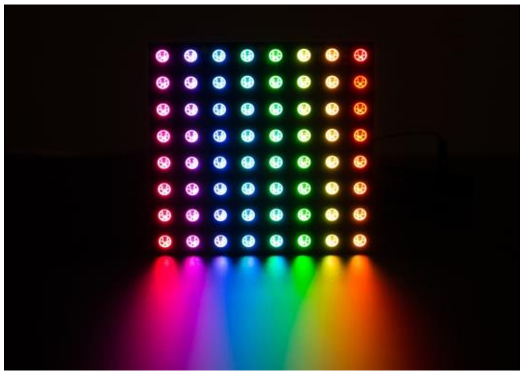


Figure 4. An 8x8 led matrix. This could be a single bead in the display for the arcade cabinet. Source: URL

For presenting a Perlenspiel project, each bead in the grid would be simulated by a NeoPixel matrix. Figure 4 shows a preassembled version of an 8x8 NeoPixel matrix. Each of the lights on the panel can have its color set independently. The operation of changing the color of each pixel can be completed fast enough to have one of these segments update its color smoothly over 30 times per second. Depending on cost and size limitations, the resolution of each bead would be variable. The display medium would also determine how many LEDs were needed. If a Perlenspiel game were being shown, ideally 8x8 pixels would be allocated for each bead. This grid size is required because one of the goals of this project is to retain as many features of Perlenspiel as possible. In order to simulate glyphs and borders legibly, at least 8x8 pixels would need to be dedicated to a Perlenspiel bead.

A downside to using multiple matrixes of NeoPixels is that the more NeoPixels there are strung together, the longer it takes for data to propagate through them. The strict timing requirements for driving the LEDs and the cost of materials make this display technology an unrealistic development approach for the arcade cabinet.

3.2. LCD panels

LCD panels are screens made up of hundreds of individual pixels. These pixels are very similar to the individual elements of an RGB matrix. However, LCD panels come in array sizes much larger than what is possible with individually-addressable LEDs.

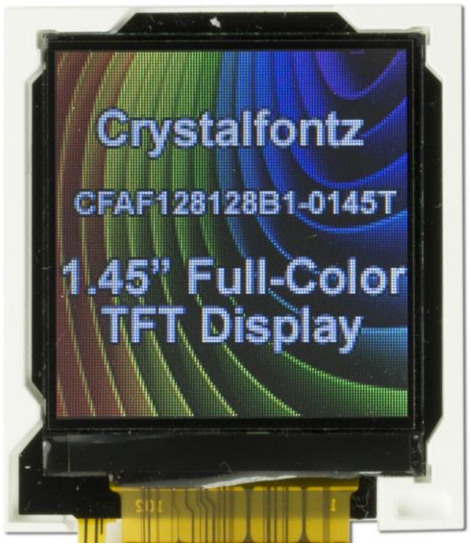


Figure 5. Crystalfontz 1.45" Full-Color TFT Display. This could be a single bead in the Perlenspiel display
Source: URL

One such panel is the CrystalFontz 1.45" Full Color TFT LCD Display (CrystalFrontz). This display has a 1:1 aspect ratio and 128x128 pixels per screen. If these displays were to be used to implement the features of Perlenspiel, the self-imposed requirement of having at lease 8x8 pixels per bead would only require the use of 4 screens. With these screens being so cheap and small, the desired 8x8 pixels per bead could be scaled upwards to 128x128 pixels per bead. This higher resolution would improve glyph and border clarity tremendously and additionally support the engine's PS.radius command to turn beads from square to circular. Using many of these screens in a matrix allows for creative demonstrations not limited to Perlenspiel. The fact that there are multiple displays could be a key feature in games designed around this idea.

The main disadvantage of this design is that driving 64 individual displays would be difficult. Each screen would need its own stream of serial data. There is also no easy way to detect input from the player; such as hand position or motion.

3.3 Ferrofluid

Another display option we considered was ferrofluid, an aqueous solution containing suspended magnetic solids. The presence of these solids allows the fluid to be shaped by the application of a structured magnetic field. Ferrofluids are normally "by volume about 5% magnetic solids, 10% surfactant, and 85% carrier." (Magnetic) Because of this makeup, ferrofluids "stand up along magnetic field lines, forming an array of spikes." While they have a practical purpose, such as flexible mechanical sealing or improving speaker sound quality, ferrofluids have also been used in a variety of aesthetic displays, like clocks or desk ornaments.

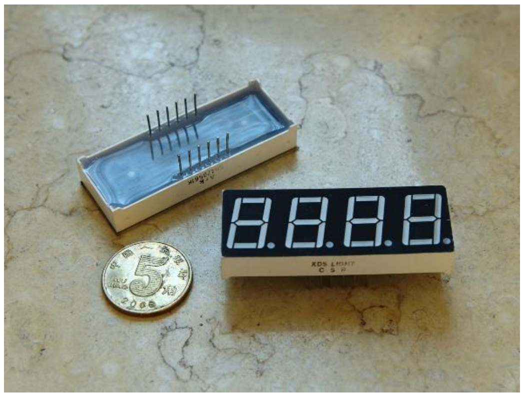


Figure 6. Seven Segment display. This shows how many symbols can be made from a smaller domain of lights. In the display a large domain of bead permutations could be made from several segments.
Source: URL

Because of their magnetic properties, with an array of electromagnets and a microcontroller, ferrofluids can be manipulated to form animated images on a 2D plane, creating an eye-catching display to passersby. Ferrofluid clocks are typically arranged like a seven-segment display, except instead of turning an LED on or off, the segments are magnets, which pull the fluid into the shape of the desired digits. These may be electromagnets or, in the case of the Rhei clock, be regular magnets moved mechanically behind the panel.

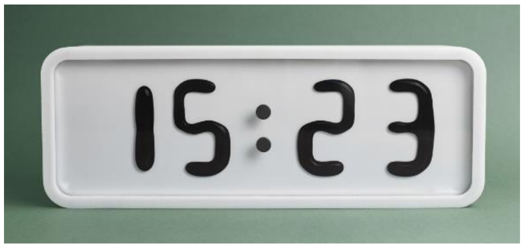


Figure 7. Ferrofluid Clock; Magnetic fluid behind glass with an array of magnetic coils form various numbers.
Source: URL

Figure 6 depicts a traditional seven-segment display in which each of the segments is an LED driven by a data line. Figure 7 shows a seven-segment display made with Ferrofluid. Here data lines cannot simply be driven to move the fluid to the desired segment, Instead, a matrix of magnetic coils must be sequentially modulated to “steer” the fluid to the desired location(s) on the clock face.

Manipulating ferrofluid is challenging, as the location that the fluid currently occupies on the display must always be kept in mind. Careful sequencing is needed to smoothly animate the fluid from one position on the screen to another. There is also only so much fluid available at one time. If the time 11:11 is displayed on a clock, all fluid needs to be evenly distributed across the area occupied by the four 1s. Those digits might appear bulky compared to others, such as 8, which require proportionally more fluid to render.

It is possible to employ ferrofluid in both 2D and 3D displays. In a 2D display, ferrofluid is more than capable of displaying a simple game like Snake or Pong, which fits nicely with the goal of offering a striking but short-term interactive attraction.

A 3D display would be much more challenging, both to construct and to design games for, but the visual impact of a floating 3D ferrofluid display would be substantially greater. By clever manipulation of electromagnetic fields, it is possible to suspend “globs” of ferrofluid in midair and manipulate their shape. Games designed for such a device would need to be kept very simple because of the physical limitations of the medium.

While ferrofluid displays are unusual and enticing, the high cost of creation and difficulty of designing for the medium make them seem impractical for this application.

3.4. “Holograms” and Pepper’s Ghost

According to Merriam-Webster, a hologram is “a three-dimensional image reproduced from a pattern of interference produced by a split coherent beam of radiation.” This precise definition has been subjected to vernacular corruption since the first true interferometric holograms were created in the early 1960s. The term “hologram” is now commonly (and erroneously) applied to almost any 3D image that can be viewed without glasses, including those produced by the HoloPlayer One.

So-called Pepper’s Ghost illusions are another example of 3D images almost universally misidentified as “holograms.” By positioning a semi-transparent mirror at a 45-degree angle between a viewer and a real object, the object can be made to appear “floating” behind the mirror. The effect was first noted by Italian scholar Giambattista della Porta in 1584. It was then rediscovered and given its current name “in the mid-1800's by British scientists Henry Dircks and John Pepper; the former invented it, and the latter popularized the effect by making it easy to perform on a theater stage.” (Wong)

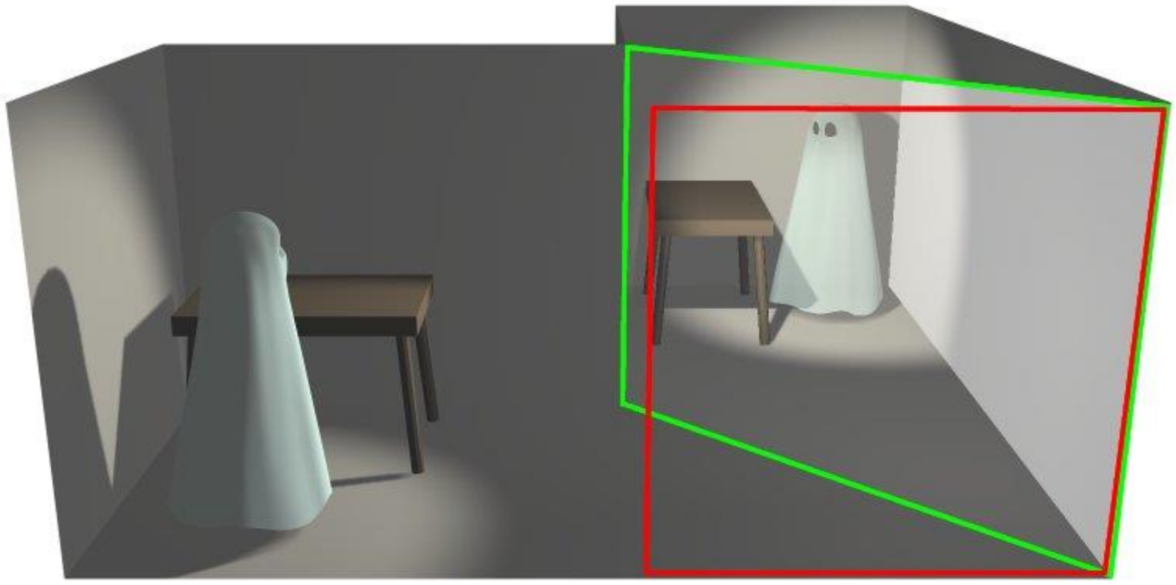


Figure 8. An illustration of a Pepper's Ghost illusion. This yields a convincing effect when well-presented, but is limited to producing images that appear behind the screen, not in front of it. Source: URL

Attractions based on the Pepper’s Ghost principle have been employed by theatrical productions and magic shows for well over century, and have been a part of Disneyland’s Haunted Mansion ride since 1969, where they are misleadingly described as “holograms.” More recently, the effect was used to simulate “live” performances of dead musicians, such as Tupac Shakur in 2012 and Michael Jackson in 2014.

Despite their apparent potential for resurrection, Pepper’s Ghost effects lack a capability that we consider crucial for our project’s purposes. We want to project images into the space in *front* of a screen, not behind it.

Actual holograms could produce the effect we want. The first holograms were composed of photographic plates exposed to the complex wavefront produced when coherent light from a laser interferes with coherent light scattered from a real object. Holocenter, a website dedicated to the holographic arts, describes the process as follows:

> "A hologram captures the interference pattern between two or more beams of coherent light (i.e. laser light). One beam is shone directly on the recording medium and acts as a reference to the light scattered from the illuminated scene … The hologram captures light as it interests the whole area of the film, hence being described as a 'window with memory.' By contrast, a photograph captures a single small area 'aperture' of perspective, the photographic image being created by focusing this light onto film or a digital sensor." (What Is Holography)

While laser holograms are capable of producing a variety of eerily vivid 3D effects, they cannot yet be used as the basis for a dynamic, interactive display. A digital interferometric screen would require pixels several orders of magnitude smaller than those used in current computer monitors. Nanopixel holography is an active area of research, but practical implementations are unlikely to become widely available for decades.

3.5. Lenticular imaging and lightfields

Lenticular imaging, which uses an array of lenses to create the illusion of 3D, seems to be the most promising technology for our project. The term “lenticular” may be unfamiliar, but most people are well acquainted with lenticular pictures. Have you ever held one of those gift cards where the image changes as you tilt it left and right? That’s a lenticular picture, a structured image mounted beneath an array of lenses which deflect light such that different viewing angles produce different images. The technique can create an illusion of depth, or used to create simple animation. It is even possible to combine both effects in a single image.

A simple lenticular image is capable of displaying two separate images, depending on the viewing angle. As visualized by Figure 9, the two images are sliced into vertical strips, and the strips are interlaced with each other. They are then placed behind a film embossed with a vertically-oriented array of hemispherical lenses. When viewed through these “lenticles,” one image appears when viewed from the left side of the array, and the other from the right side. Adding additional interlaced images allows animation effects when the image is tilted.

If the interlaced images represent the left and right sides of a stereo pair, the lenticles can be used to “steer” each image to the appropriate eye of a viewer, creating a 3-dimensional effect. Autostereoscopic lenticular images can include elements that appear either behind or in front of the image plane, or both simultaneously.

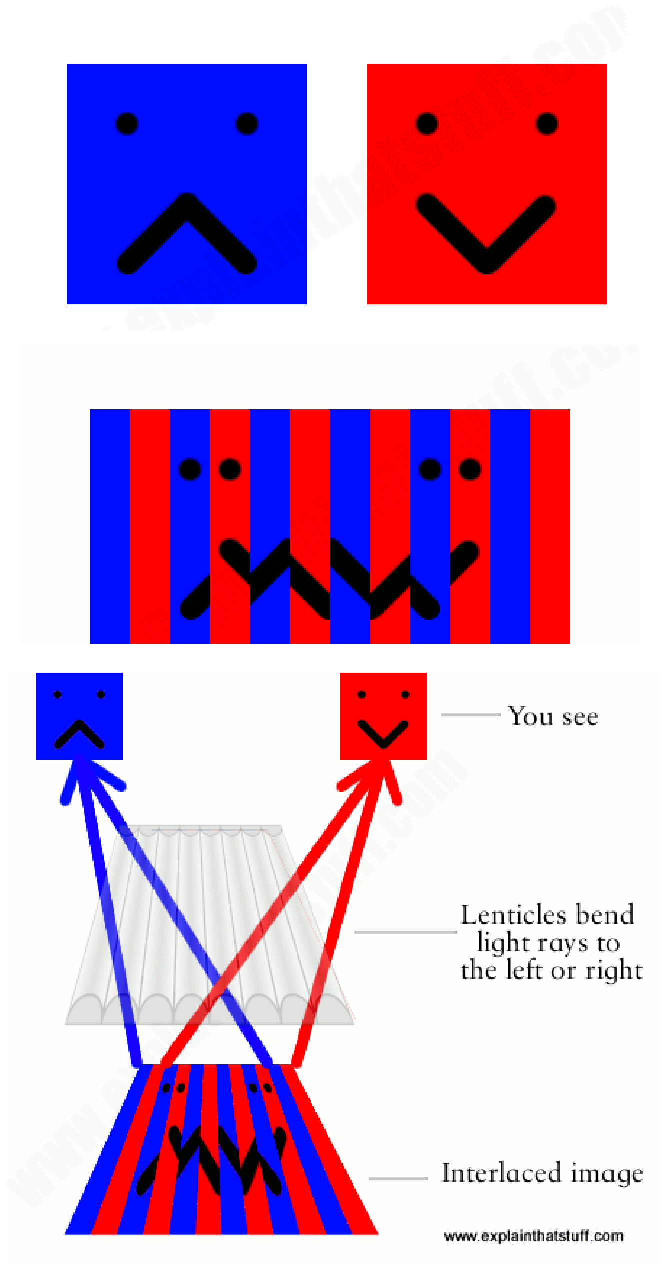


Figure 9: A depiction of how an interlaced image can be separated with lenticles to form an independent image for each eye. The arcade cabinet could use this technology to create a 3D image for the viewer, without glasses. *Source:* URL

Lenticular pictures are an elaboration on an earlier technology called a "fly's eye lens array," proposed by Professor Gabriel M. Lippmann in 1908. This was an array of many small convex lenses (arranged in a pattern similar to its namesake), which could "record a complete spatial image with parallax in all directions." (Roberts)

Both the Lippmann array and its lenticular offspring are crude examples of *lightfields*, 3D images formed by optically multiplexing a large number of individual views. Augmented reality systems under development by companies such as Magic Leap are reportedly based on digital lightfield technology.

Despite its misleading name, the HoloPlayer One device discussed previously does not employ true holography. It is actually a sophisticated lenticular display that uses a computer monitor instead of a printed image to produce the interlaced stripes required to yield a 3D effect.

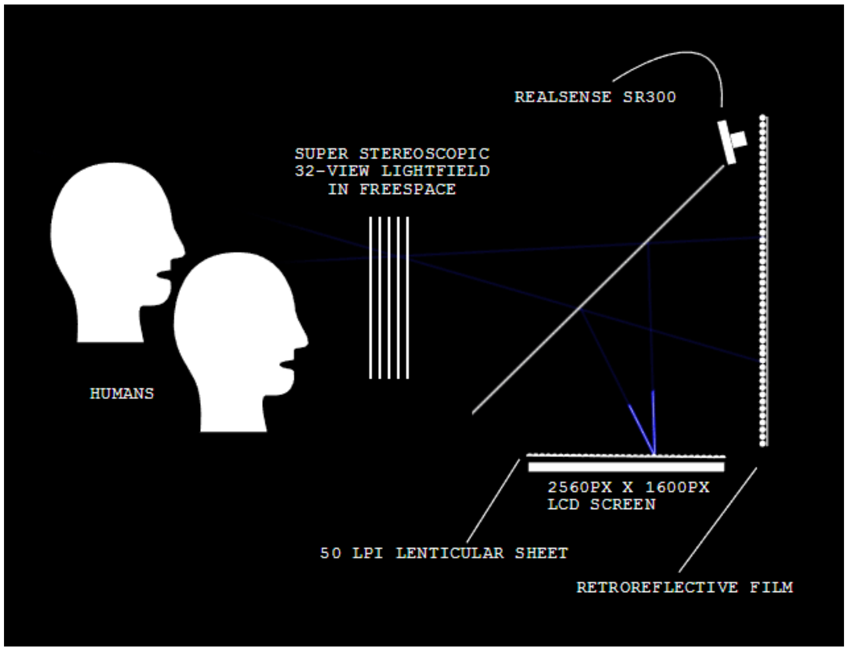


Figure 10. A diagram showing how the HoloPlayer produces an autostereoscopic display for up to two simultaneous viewers. *Source:* URL

As shown in Figure 10, a 3D scene is rendered from 32 horizontally-displaced viewpoints into separate images, each 267 pixels wide and 480 pixels high. (Higgins) These 32 images are decomposed into slices and sequentially interlaced for display on the monitor.

Light from the monitor is transmitted through a 50 lines-per-inch lenticular sheet, then reflected 90 degrees through a polarized beamsplitter onto a retroreflective film, producing a 3D image that appears to be suspended in the space between the beamsplitter and the viewer.

The geometry of the display insures a smooth transition between images as the viewer's eyes move horizontally across the display zone, which is wide enough (barely) to accommodate two people simultaneously.

The main constraint of this technology is the limited field of view and low resolution, as a result of splitting up a 1440p screen to simultaneously display 32 images. As a result, the available design space for the final product is limited to games which don’t rely on highly detailed imagery. This is more than achievable, using either the provided SDK for the Unity game engine or using a custom version of the Perlenspiel engine, using the appropriate interlacing shader. Due to the limited field of vision, we’ll be mounting the provided kiosk inside an arcade cabinet, which also provides other benefits, further detailed in the Propositions section.

3.5 Motion controls

While a traditional arcade machine usually has a joystick and 2 to 4 buttons, the HoloPlayer One uses an Intel RealSense camera mounted above the beamsplitter to detects hand motion, allowing users to interact with the displayed image. (Timmer) While either of these approaches would be suitable for the display, having both would clutter the interface and give the player mixed signals. If the display has strong signifiers for motion controls, the physical controls might be ignored, and vice versa. Therefore, we have to choose one or the other.

Having physical controls at first seems to be the most natural solution. Most if not all of our potential players have experience with arcade machines and their traditional input devices, joysticks and buttons. This familiarity offers a number of benefits. Because players will recognize the controls, they will (correctly) assume that the device is interactive, making the controls an effective signifier. Also, their prior experience with such controls means that they will need minimal instruction to understand how to interact with the display, both freeing up the space that would have been used for instructions and lowering the time it takes to understand the game.

However, the motion controls provided by the HoloPlayer One are potentially far more compelling. While players will have much less experience with such controls, there is enough common knowledge that, assuming the software is designed with simple gestures in mind, players will be able to adapt to them fairly quickly. Additionally, though the change may seem simple, moving to a motion control system has been shown to increase the overall effect and immersion of the experience. (Birk)

Aside from this, motion controls contribute to the overall theme of our exhibit, playing on the connections popular culture has established between “holograms,” futuristic technology and science fiction. While an autostereoscopic display is inherently intriguing, a player may feel disappointed if they have to use old-fashioned physical controls to interact with it. In order to fully bring to life the Star Trek fantasy, motion controls are the most viable option for making an appealing display.

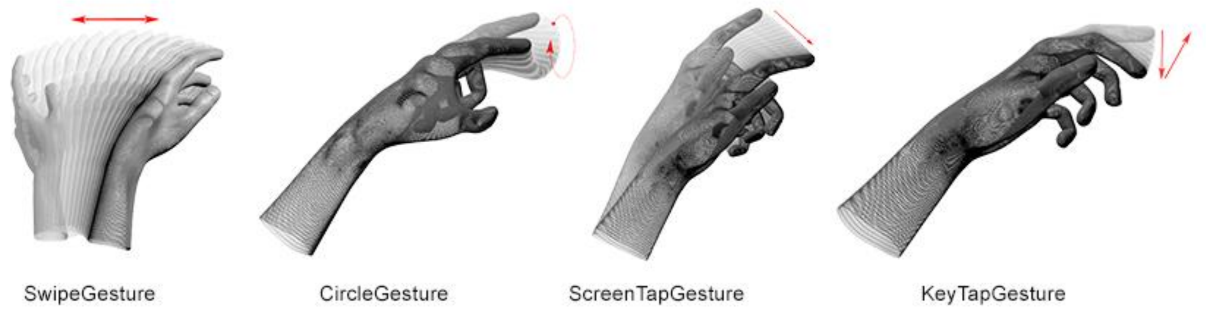


Figure 11. Sample Motions for the Leap Motion Controller. These gestures could be interpreted as new Perlenspiel events which have their own callbacks. Programmers could then control the Perlenspiel game state using Gesture controls. *Source:* URL

Because of the short time our player will spend interacting with the display, there must be relatively few controls to learn, and what controls there are must be intuitive. The developers at Leap Motion, producer of the eponymous hand-tracking controller, define an "intuitive" interface, with or without motion controls, by three criteria: "an interface must be learnable, understandable, and habitual." (Plemmons)

In our context, learnability and understandability should be synonymous. The user should not have to learn much further than their initial understanding to get the full experience of our games. Likewise, habituality is less about building new habits with our interface as it is utilizing habits our users likely already have.

But how do we rate the quality of our controls? In 2014, Minna Hara and Saila Ovaska of the University of Tampere in Finland wrote an article establishing 13 heuristics for qualifying motion-based controls in games. (Hara) Some heuristics focused on the motions of the gestures themselves. Gestures should be sufficiently distinct from one another to minimize confusion between them, and also distinct enough from idle movement so as to avoid false positives. They should also match realistic, natural mappings for movement, while avoiding being intense enough to tire the player out. Others focused on the game's reaction to the gestures. When a gesture is executed, the game should give clear feedback to indicate as much. Some other heuristics, like Space and Multiple Players, aren't a concern for this particular interface, as the former is already provided by the cabinet housing and the latter is either discouraged or not a focus of the project. Future projects for this display outside of the results of this project should take these heuristics into mind, as a shoddy interface could discourage players from interacting with the display.

As mentioned earlier, gesture controls also strongly suggest science fiction, fitting the theme of the display technology as well. Christopher Noessel, a head of UX design at IBM, wrote an article on Gestural Interfaces and how we can learn from what we see in science fiction, focusing mostly on the 2002 Tom Cruise film *Minority Report*. (Christopher) Most of his example gestures are also things we see in touchscreen interface design: pinch to zoom, swipe to dismiss, and touch to select. However, he strongly emphasizes the importance of these gestures and the core principle shared between them: that the user should manipulate whatever they control as directly as possible. While on a computer, one might scroll through a page by using the mouse to move a scroll bar, on a touch screen, the user drags the page up or down directly.

Noessel also points out an interesting supplement to touch and gesture controls: natural language processing. Because there are some actions that cannot be effectively encapsulated by a series of gestures, we might consider enabling the display to process simple commands. As mentioned earlier, the nature of this display evokes science fiction. Having a voice processing system that would bring to mind *Iron Man's* JARVIS or *Halo's* Cortana would sell that kind of feeling. However, voice control introduces a number of potential problems. While natural language processing has made great strides in recent years, it's not always consistent, and inconsistency hampers the user's experience. Commands would need to be very simple in order to minimize error.

Additionally, this display will be in a public space. Users are less likely to use voice commands in public so as to not disturb other people, and those commands are less likely to be heard properly depending on the amount of noise produced by those same people. Nonetheless, if we overcome these issues, voice commands may be a useful option.

One way to do so would be an alternative to natural language, while still using vocal input. A study done by a group at the Czech Technical University in Prague indicates that humming is a strong alternative to natural language. (Sporka) By detecting variances in pitch, timbre, and volume, humming or even whistling is a useful source of voice input. One advantage to this is a lowered delay in

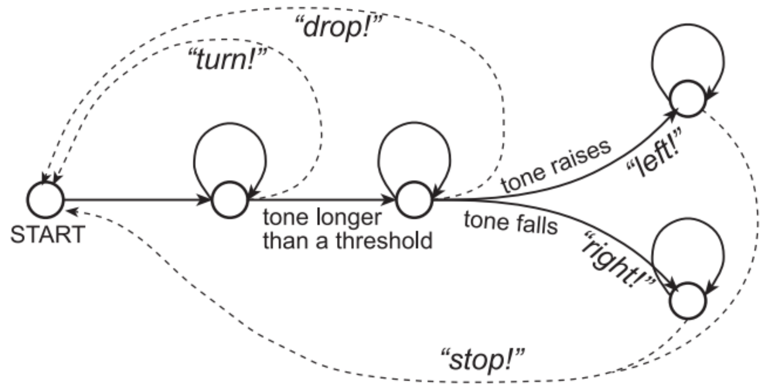


Figure 12. The tonal gesture system used by their team

response time, as users aren't constrained to the exact length of a word, and can change their pitch on the fly. In fact, the team measured that humming had a higher effective speed, or cells moved per second, than speech controls in the game *Tetris.*

Despite this encouraging research, there are drawbacks to this approach. The study mentions that subjects were given 15 to 30 minutes to adjust to the control scheme, a luxury we will not have due to the brief interaction periods likely to be available. Also, there will be no human present to relay the instructions, so humming-based controls will have to be listed and clearly explained in order to be noticed, especially because an arcade cabinet does not signify audio input, and in fact may do the opposite.

4. Propositions

4.1. Arcade cabinet

We propose that our team uses this technology in a public arcade cabinet, to boost the visibility of the IMGD major on campus. Students are busy, and tour groups move quickly, so a good advertisement for interactive media must be public and attractive, must elicit play, and must be available for brief interactions. We have determined that an arcade cabinet well suits these needs, as there are a variety of places where this display would both fit in and stand out. There are plenty of resources to make this exhibit look great, the design elicits play on its own, and the game or toy that will take up the exhibit can easily be designed to suit both short and long-term experiences.

There are three locations that seem most suited to host this exhibit: the Rubin Campus Center, Fuller Laboratories, and Salisbury Laboratories. While the campus center is the hub for a greater variety of activities for people of all backgrounds and interests, Fuller and Salisbury are the two buildings which actually host IMGD facilities, and are the two buildings in which IMGD classes are usually taught. For the purpose of tours, this wouldn’t make much of a difference, as tour groups generally stay in any given location for roughly the same amount, but for students, the decision is much more important. People visiting WPI outside of tours, such as alumni or investors, are much more likely to visit the campus center than either of the lab buildings, which might take priority over getting more student interest.

The most visible feature of an arcade cabinet, aside from the game itself, is the design on the paneling. The combination of the flashing images on screen and the artwork on the sides and front have been drawing players in since the 1980s. While the game inside won’t necessarily be a callback to the 80s, the art is no less important. Because of this, our team will likely include an IMGD art student for that express purpose. The art can either be painted on the panel, or we could print out the side panels through the WPI printing services or in the Washburn machine shops. The majority of classes a student will take are directly within their major, reducing the number of people the average student interacts with to the same people. This is an opportunity to incorporate students with multiple majors into one project, paving the way for innovation.

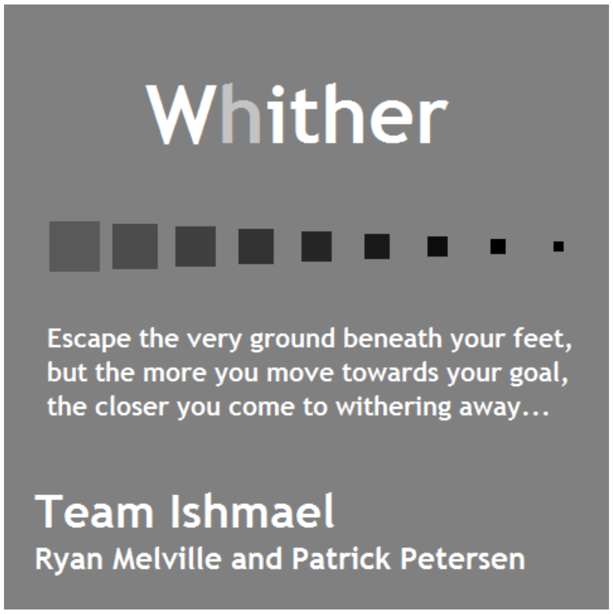
Aside from the art, simply the shape of an arcade cabinet is also enough to get players interested. The bright buttons and joystick suggest the ability to interact and manipulate them, everyone knows what an arcade cabinet is and how to use one.

Taking this into account, the cabinet itself needs to avoid leading people astray with their own ideas of a "traditional" arcade cabinet. First, it must be apparent at a glance that the machine does not take any money or token to play. It's possible that a potential player might see the cabinet and walk past without further examination, since they don't have any quarters on them. Secondly, if the cabinet runs a toy instead of a game, it must be clear that the toy does not have a "win condition," as the player might end up getting frustrated that they can't win.

Finally, it must be clear that the game or toy will not take up a large amount of time. Students are busy, and while many love distractions for long periods of time, others are wise enough to avoid them like the plague. As such, the cabinet and game must somehow indicate that prolonged interaction is optional. Depending on the software installed, this could be variably difficult.

Figure 13 An arcade cabinet Link

Unfortunately, the HoloPlayer One features a hand tracking system for its default control scheme, which does not have a plainly visible signifier. There are a number of possible solutions to this problem. The simplest is to remove the normal arcade controls and assume that players will naturally try to touch the floating image. While this method would likely be somewhat reliable, it may also have the opposite effect: no visible controls means it can't be interacted with, therefore it must be a do-not-touch museum display. As such, it will ultimately be necessary to provide some sort of signage, either to indicate that the display can be interacted with, allowing players to discover the interactivity of the image, or specifically that the "hologram" can be touched. This is the clearest approach, but also may take away a moment of wonder when the player realizes they can manipulate this very sci-fi floating image.


Whither
Escape the very ground beneath your feet, but the more you move towards your goal, the closer you come to withering away…
Team Ishmael
Ryan Melville and Patrick Petersen
Figure 14. Due to the nature of light, even great games like Whither may not be compatible with the arcade cabinet, as the holographic nature doesn't do well with a desaturated palette. Source: Link

On that note, because of the restrictions on memory and the maximization of profit, arcade games were typically impossibly difficult to beat in one go and required multiple 'retries' to beat, often based on inserting more money or tokens. Therefore, the hardest part of making an arcade machine is convincing players that they can come back later. While this largely rests on the software component of the machine, there are some small but significant changes that will draw players in without anchoring them.

First, the room the machine will be housed in will be brightly lit, unlike an arcade. Of course, arcades afford play, making it clear that this object can be interacted with in the interest of fun. Additionally, this will be a standalone machine. Without the atmosphere of people playing games, players are unlikely to stick around for too long. Aside from hardware, the software itself must also be designed to afford short sessions. If we do decide to use Perlenspiel, that will come in handy, as the engine’s graphical design aids the development of simple games that convey their experience in a quick and straightforward manner.

4.2 Custom engine

Rather than rely on the Unity implementation of the HoloPlayer’s display driver, it would be impressive to create a custom application which would interpret Perlenspiel games and display them on the HoloPlayer device. This application would need to be able to send graphical data to the HoloPlayer, receive player positional data from the HoloPlayer, receive game data from the internet or a local file, and interpret Perlenspiel.

Many libraries exist to help applications complete these tasks. Libraries such as the Open Graphics Library (OpenGL) make rendering complex 3D scenes possible at very fast speeds. Libraries such as JSerial (Hedgecock) allow for serial data to be sent out of or read into a computer over a specific serial port. The Nashorn Scripting Engine (Nashorn) allows for easy interpretation of JavaScript. Computer languages such as Java were made to integrate with the web and local file system. The combination of these libraries, and the Java computer language would allow for a custom application optimized for the interpretation Perlenspiel games to be created.

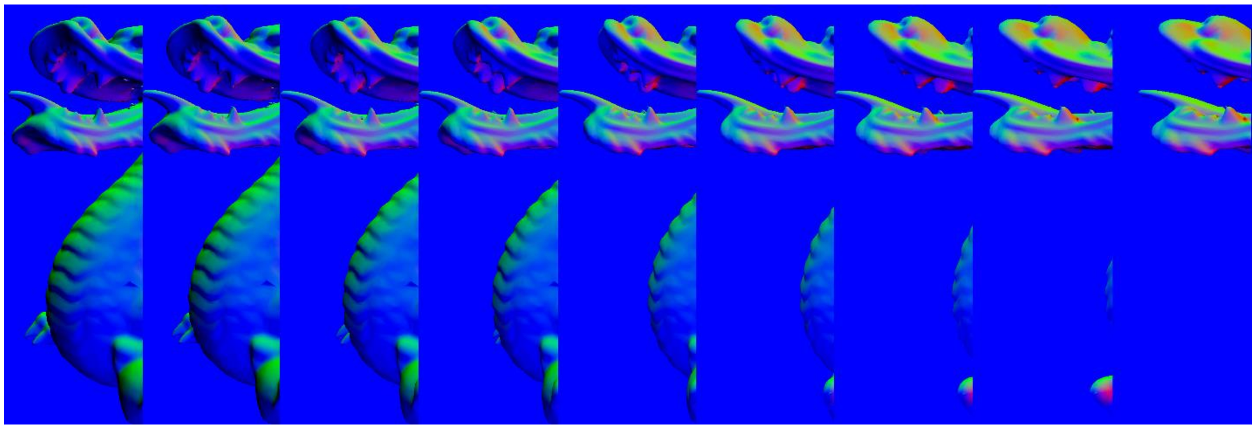


Figure 15. A representation of how different camera angles change what part of a model is seen. The left side of this image is a front view of the Stanford Dragon model, where the right is a side view. This image was made for this paper by Bailey Sostek, the model file is linked below. Source: URL

The interior of the arcade cabinet will house a computer. This computer will have a graphics card which can process commands with OpenGL. The HoloPlayer needs to receive data in the form of a compressed 32-view image. Each one of these views contains a version of the scene being rendered, however each view of the scene is "taken" from a slightly different angle. This could be mimicked with a simple geometry of 32 rectangles, each textured with a view of the main scene at a slightly different angle. This would be achieved by placing cameras around a 3D scene, and mapping what each camera sees to the 32-view grid.

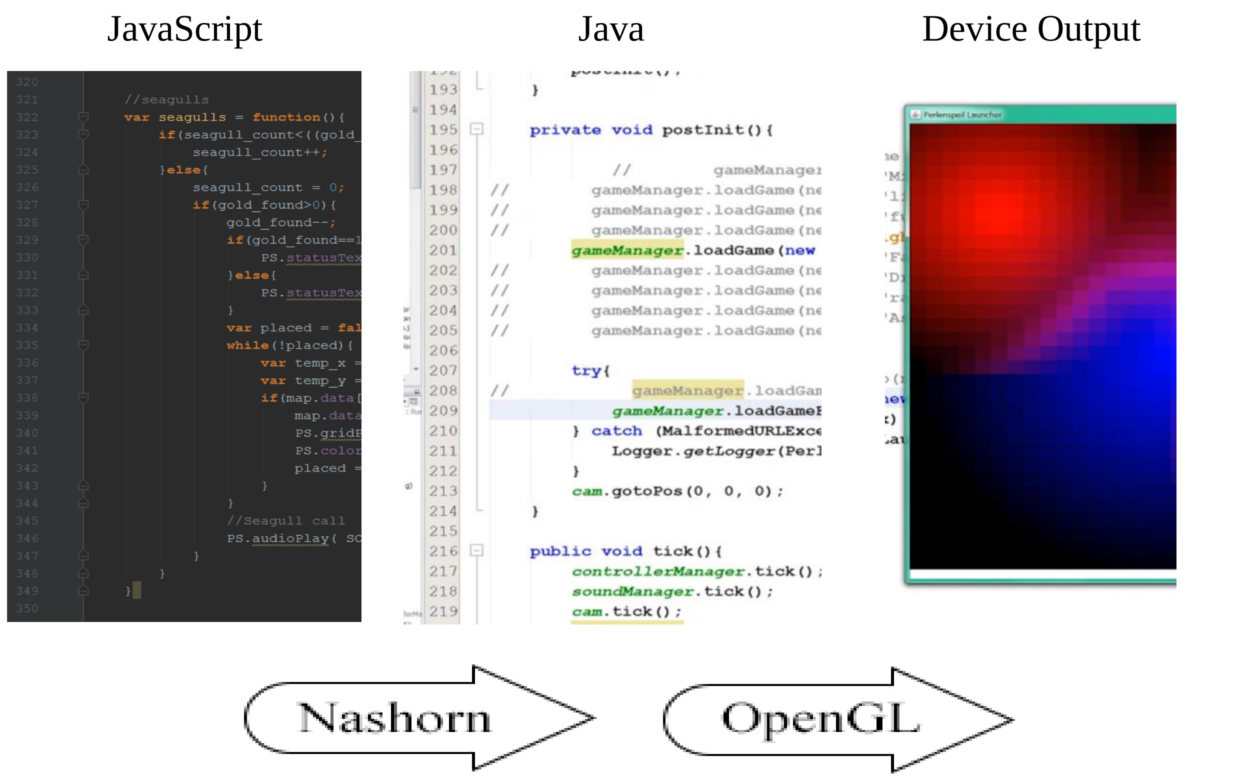
Figure 15 shows what this would look like visually. The image is comprised of nine views of the Stanford Dragon Model, all from slightly different angles. Each one of these images would show selectively, based on the angle the viewer is observing the HoloPlayer at. The most similar view will be selected and shown, creating a pseudo-3D effect. This image would be created through the help of hardware acceleration to allow for these images to be generated and chosen many times per second.

The Nashorn Scripting Engine provides classes such as ScriptEngineManager, which allows a programmer to specify the type of script files that that engine should interpret. If the parameter “JavaScript” is passed into an instance of this class, the class can then be passed JavaScript files to be interpreted. The interpretation encapsulates all functions built into JavaScript, such as the Math class. Any functions the engine does not know about will be thrown as exceptions when these unknown functions are attempted to be interpreted.

When a Perlenspiel function is interpreted by the ScriptEngineManager, the default events that the Perlenspiel web library knows about cannot be interpreted. There is nothing locally to define these functions, and tell Perlenspiel how to display the data. All of the events shown in Figure 3 need to be mapped to new functions. This can be achieved very simply through using the ScriptEngineManager function “put()”. This allows for a string of text to be mapped to an internal Java function. In the case of the first Perlenspiel event, PS.Init, the string “PS.init” can be mapped to an internal java call to initialize the arcade cabinet and get ready to display bead data.

In order to encapsulate all functionality of a Perlenspiel game, the facilities provided by the Perlenspiel API would need to be redefined using the Nashorn “put()” function. Remapping all functions in this way would create a Perlenspiel emulator, where all engine calls are captured and remapped to have the same effect that they would have on the web.

The Perlenspiel engine itself dynamically modifies the HTML on a webpage in order to create a visual response. The custom engine would be running in Java. Therefore, there would be no way to modify HTML to show that the engine was working. A Java implementation for setting the color of specific squares on the display would need to be created. This function would then be mapped to the call PS.color(). Any rendering capabilities need to be handled through the custom engine in this way; such as setting the glyph color, or background color. Rather than displaying these changes in HTML, the graphics capabilities could be anything from a computer monitor with a window open, to an LED panel being sent serial data. The only requirement that the display technology has is maintaining a grid that can be modified with Perlenspiel commands. This grid data will be stored in memory and can be referenced through Java and handled as is desired.



```

    //seagulls
    var seagulls = function(){
        if(seagull_count<((gold_
            seagull_count++;
        }else{
            seagull_count = 0;
            if(gold_found>0){
                gold_found--;
                if(gold_found==1
                    PS.statusTex
                }else{
                    PS.statusTex
                }
                var placed = fal
                while(!placed){
                    var temp_x =
                    var temp_y =
                    if(map.data[
                        map.data
                        PS.gridP
                        PS.color
                        placed =
                    }
                }
                //Seagull call
                PS.audioPlay( SO
            }
        }
    }

```

```
        postInit();
    }

    private void postInit(){

            //          gameManager
//        gameManager.loadGame(ne
//        gameManager.loadGame(ne
//        gameManager.loadGame(ne
        gameManager.loadGame(new
//        gameManager.loadGame(ne
//        gameManager.loadGame(ne
//        gameManager.loadGame(ne
//        gameManager.loadGame(ne

        try{
//              gameManager.loadGam
            gameManager.loadGameF
        } catch (MalformedURLExce
            Logger.getLogger(Perl
        }
        cam.gotoPos(0, 0, 0);
    }

    public void tick(){
        controllerManager.tick();
        soundManager.tick();
        cam.tick();
```

Figure 16. A depiction of several languages determining how Perlenspiel is rendered. This is a custom engine which interprets JavaScript Perlenspiel games, to produce the same output in a window, rather than a web canvas.
Source: URL

Figure 16 shows an example of how code will be interpreted between different languages. The arrow running from JavaScript to Java shows that Nashorn will be used to interpret JavaScript and integrate with Java. The arrow running from Java to Device Output shows that OpenGL will be used to interpret and render the graphical data being processed in Java. This last part of the process can be switched out for anything. In the example shown in Figure 16**,** the raw data stored in Java is being interpreted to become a graphical display on a windowed canvas. This windowed canvas could be replaced with anything, such as sending out packets of serialized data, or writing bead data to a text file.

There are many benefits to creating a local interpreter in this manner. First of all, when Perlenspiel is emulated locally, the CPU can handle all computational math, while the GPU interprets and displays the visuals. Applications that run on the web are hosted through a web browser, which slows down the speed at which a Perlenspiel game can be interpreted. With a local interpreter this bottleneck does not exist.

Another benefit of local interpretation is the ability to add additional commands. Since JavaScript is a scripting language, functions are compiled at runtime. This means that functions inside of 'if' statements are not evaluated unless the 'if' statement is evaluated to true. This can be exploited by setting the Perlenspiel engine variable for the platform that the games is running on to something arbitrary, such as "Custom_Engine". Then "Custom_Engine" specific calls can be hidden within 'if' statements checking for 'platform === Custom_Engine'. The only case where 'platform === Custom_Engine' evaluates to true, is the custom interpreter, where the internal new functions are also defined. For instance, if a new command setBacklight (Red, Green, Blue) were defined, it could only be called on the custom engine by putting this new command inside of an if statement checking for 'platform === Custom_Engine'. On the web, nothing would happen, however the local interpreter could evaluate this call to true and then run the internal functions. This would enable additional flexibility to interface with an arcade cabinet that the initial game engine did not account for, such as an attract mode, or loading and unloading Perlenspiel games on a win or loss.

Once a custom engine has been developed, it can be used to stream Perlenspiel games from the web. Java has a built-in class called URL which takes in a web address and returns a connection to the data contained at that location on the web. This data can then be streamed from the web to the Java program by calling openStream(). If the URL to a Perlenspiel game's JavaScript file was passed into a URL object, that JavaScript file could be streamed directly into the Nashorn Scripting Engine, then interpreted locally. This increases the amount of student-made work that can be shown on the console. Games can be directly featured on the console itself by storing games in the console's local file system and games can be streamed through the web. Any student who has ever made a Perlenspiel game and put it on the web could stream their game directly to the console to show it off for all to play.

A goal of the custom engine approach will be to mimic the visual appearance of actual glass beads such as professor Brian Moriarty described in his GDC talk**, "**Originally I was going to make them look like real glass beads…**"** (Moriarty Lehr) This would be a great demonstration of a local interpreter working. All beads would react and behave in the same way they did on the web; however, they would appear to be glass beads in the local interpretation. This would be achieved though replacing the default rendering of the beads as textured quads, with reflective and refractive Taurus shaped 3D models.

4.3 Unity approach

The core of our system, the HoloPlayer One, comes with an SDK to be used with Unity. Unity is a cross-platform engine used in games worldwide. Over 80% of games on the App Store are made with it. Though written in C/C++, designers program software in Unity entirely in either C# or (much less frequently) JavaScript.

Using the provided HoloPlayer One SDK, Unity presents an easy, natively supported way to create a game or toy for the display. Not unlike Perlenspiel, Unity is more than accessible for student game designers, so that IMGD majors can use this display to show off projects. While the resultant design space may be constrained by the fixed controls and the low resolution, these constraints only offer more opportunities for design.

Aside from games and toys, Unity could also be used to display the 3D models made by IMGD art students and the 3D modeling classes, in an interactive setup. Not only does this have a distinct 'cool factor' (who wouldn't want to see the character they made projected in front of them in real life?), it also serves the practical purpose of letting the artists naturally view their art from any perspective.

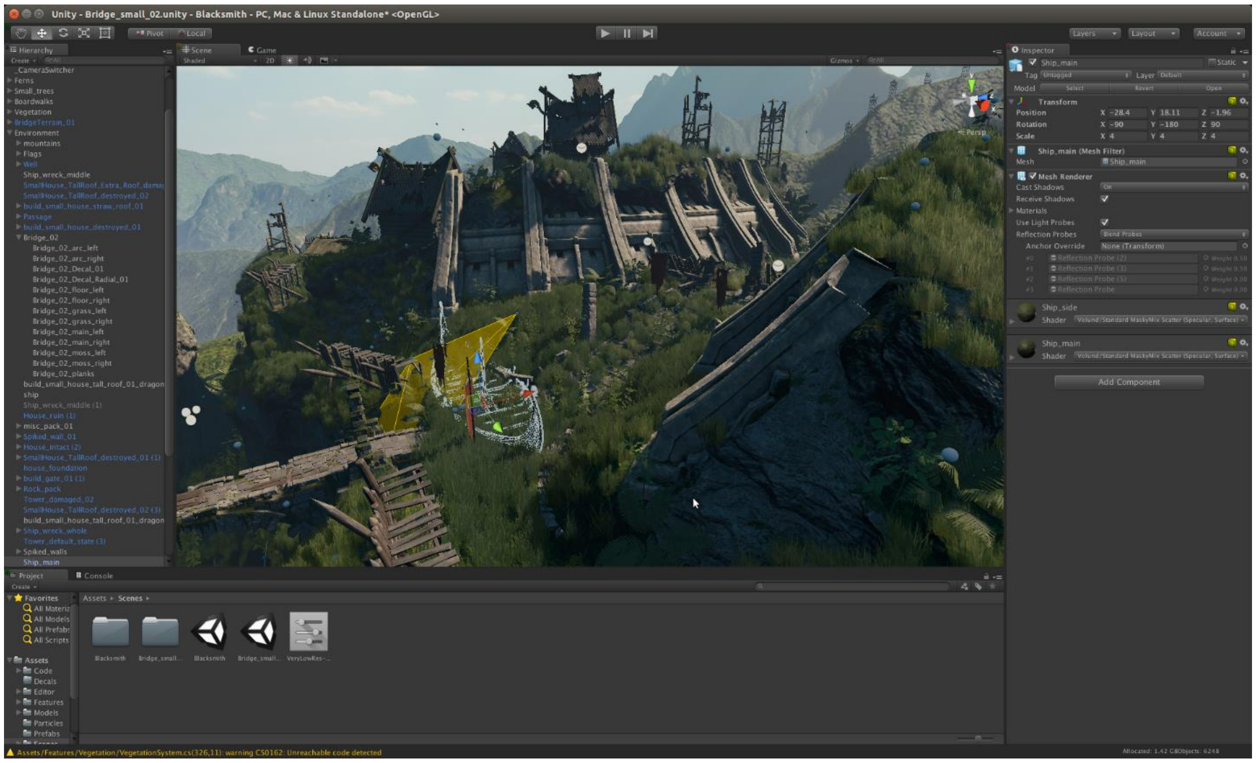


Figure 17. The Unity Editor showing how easy it is to create and manipulate world objects. Perlenspeil could be scripted and played in the Unity Editor very simply.
Source: URL

Incidentally, we need not decide between Unity and Perlenspiel. Because of Unity's support for JavaScript, instead of customizing Perlenspiel to fit the HoloPlayer One, we can instead have an implementation of Perlenspiel which executes a standard Perlenspiel game within a Unity game. This has several advantages. First and foremost, the required work is significantly less. Managing a grid of beads as an array of game objects is a trivial task, especially given that the next version of Perlenspiel is being written with the graphics and logic separately. Next, Unity allows much simpler control over the aesthetic that this Perlenspiel port would have. The size and appearance of the base bead object could be manipulated with the built-in material and lighting support. In fact, we could even give designers the ability to manipulate these materials in Unity through the provided JavaScript, to add another layer of options to their game designs.

4.4. Game vs toy

The next question that needs to be asked: are we making a game or a toy? A toy is defined rather simply as a thing that elicits play. A toy invites you to spin it, a sandbox invites you to manipulate the sand, etc. What distinguishes a game from a toy is a ruleset and one or more goals. A hacky sack is a toy, the eponymous game is a game because of the rules and goals involved: keep the sack in the air, and you can’t use your arms to do so. In the case of this display, this distinction offers two very different paths in our design.

Making a game at first appears to be an intuitive solution, after all, this is to show off the Interactive Media and **Game** Design major. This offers an advantage in and of itself, as a game in a display on campus would immediately bring to mind the game design major on said campus. However, there are some drawbacks to keep in mind. While not a major concern, a game with a definitive end is much less likely to be revisited once completed, compared to a toy or a sandbox game**.** Additionally, the natural desire to complete a game may leave players unsatisfied from short interactions. While that may encourage some to come back, the same lack of satisfaction may turn potential players away completely.

Comparatively, making a toy seems much less risky. Because you can’t ‘win’ a toy, the concept lends itself to short interactions. A player can pick it up for as long as they would like, and put it down with no obligation to continue. Phrased differently, players won’t be afraid to pick up the game because of time constraints, and even busier students will be willing to try it out. This also makes it better for tours, as there’s no need to ‘start’ a toy either, it should always ready to be interacted with, and reset if need be when left alone.

4.5. Biomedical

The application of making an arcade cabinet to display student made work would not need to be limited to IMGD. For instance, students and professors doing research in the biomedical field would be able to display their work on the arcade cabinet. Anatomical models, or data gathered from research could be visualized and displayed in 3D space using the HoloPlayer. Creating an application to render this information on the display could be created through an additional MQP at some point in the future, but is out of the scope of this project.

5. Discussion

5.1. Display hardware

The first decision we had to make was the most important one: what hardware would be used to run the display? At first, we looked at using LED panels, specifically with the Perlenspiel engine in mind. With a grid of these panels, we could replicate a physical version of the engine, with each bead being represented by its own individual panel. While we would have to discard the glyph and border feature with one LED per panel, we could also have a grid of LEDs in each panel to keep these features, at the expense of complexity. However, while there is a novelty in this approach, in practice, it’s not much better than using an LCD touchscreen, which would retain all the features of the engine and also require much less work to set up.

Another option was ferrofluid. Instead of using Perlenspiel, we would instead write our own engine built with the properties of magnetic fluid inside. This creates several opportunities unique to this approach. Instead of rigidly defined tiles, the fluid can move around and between tiles. Ferrofluid offers unique constraints as well: all games must be black and white, with the black being the moving fluid; because of the physical existence of the fluid, designers need to account for its moving back and forth; and finally, designers must account for or utilize the effects of gravity.

However, we ultimately settled on using the Holoplayer One. We decided that the visual aspect of an autostereoscopic display would have a greater impact. While black and white provides unique design opportunities, it also takes away from cases where color would have an effect.

Additionally, the HoloPlayer One provides SDKs for many popular game engines and graphics APIs that will better allow students to use the device.

5.2 Engine choice

The next major point of contention with the path this project will take is what engine would be used primarily to run the display. The options mentioned earlier were a custom SDK for an existing version of Perlenspiel, the Unity Engine with the Unity HoloPlayer One SDK, and the next version of Perlenspiel using the Three.js HoloPlayer One API.

Creating a custom engine for Perlenspiel would be impressive and allow for a lot of tweaking of our desired specifications, but would take much more work than other options. Additionally, the engine would need to be maintained by someone, and also be designed in such a way that it's easy to pick up, so students are encouraged to build on this platform.

Instead, it might be better to use an engine that students might be more familiar with or is easier to learn. Luckily, Looking Glass Factory has provided an SDK for both the widely popular and free Unity engine as well as an API for Three.js, which the next version of Perlenspiel is being built upon. Both engines are a compelling prospect as a result. Because WPI students are more likely to be familiar with Perlenspiel, this engine may be a superior choice. However, the limitations of Perlenspiel, while excellent in a course teaching game design principles, may limit the possibilities of a 3D display, in which case Unity would prove to be a better option.

5.3. Launcher

Instead of setting a standard for allowed engines on the device, we could simply create a launcher that would run any given executable. Of course, in order to maintain the feel of the display, the launcher itself would have to run in the display. This comes with a few complications. All submitted games and toys would have to follow a standardized format. While this wouldn’t be a problem for most games, some people might have an issue with it. Also, the interface would have to have an easy and standardized way to exit back to the game selection screen from any given game. If we were to reserve a gesture for this action, whichever gesture we picked would be unavailable for all games, which might cause a problem later down the line. We also have the option of having this exit button in hardware on the display cabinet. While this would take away from the “purity” of the display, it would also have the least potential for conflict. We could also completely separate the launcher from the displayed “hologram” and instead have the launcher run separately, on a regular touchscreen. Again, this might hurt the aesthetic value of the display, but the tradeoff may be worthwhile.

5.4 Potential course tie-ins

The specifics of the display aside, there are a multitude of classes at WPI that would be able to make use of it outside of its utility of bringing attention to the IMGD department. Our 3D Modeling classes, for example, could use the display to show off the results of their work, or evaluate them themselves. Computer Graphics classes may use the display to explore different ways of using a shader to project images in 3D. Most likely, our Novel Interfaces class may take interest in this, as this technology is likely very new to most of our students. Learning about gesture-based interfaces will be especially valuable in the near-future, seeing as software designers are already starting to make headway into the field of AR, or augmented reality.

5.5. Sample software

While the device will mainly be meant for student use, we as designers should create examples of technology to help build interest in the display in the first place, and to give other students examples to alter and build upon. In order to do this best, we need to show off the core functionality of the HoloPlayer One and create cohesive experiences with these elements. The games or toys don't need to be particularly innovative in their own right, but still show off the good side of the HoloPlayer.

For example, we could create a toy in which a 3D ball of goo hovers in the center. Players can pinch and pull the ball to make it stretch with their fingers. They may also slap it to make it break and reform. Simple but addicting interactions should pique interest in the device, and make a designer think about how they could use it better. Additionally, as this display will be in a public place, the display should also be moving even when no one is interacting with it. This may be as simple as a 3D "screensaver," or a more interesting reaction with a submitted toy, such as a plant growing on its own, or animals interacting with their environment independently.

6. Conclusion

The goal of this project is to make an attractive public display for users to interact with in order to get a perspective of the IMGD program through play. In order to expose as many individuals to as much IMGD content as possible, the device needs to be flexible in what it can display. The primary content that we want to display on the device would be student-made Unity games, as well as Perlenspiel games. The HoloPlayer One device has a Unity SDK which enables any Unity game to be implemented in 3D on the HoloPlayer device very easily. The HoloPlayer also has a library which integrates with the JavaScript library ThreeJS. This library is the graphical backend for Perlenspiel 4, making the transition from web to HoloPlayer very easy. Because the HoloPlayer has support for the two primary forms of content we want to exhibit, it is an excellent choice for the arcade cabinet.

7. Works Cited

Adafruit. “NeoPixels.” *Adafruit Industries, Unique and Fun DIY Electronics and Kits*, Adafruit, www.adafruit.com/category/168.

Birk, Max, and Regan Mandryk. Control Your Game-Self: Effects of Controller Type on Enjoyment, Motivation, and Personality in Game, ACM, 2013, doi:10.1145/2470654.2470752.

Crystalfontz. “128x128 1.45’ Full Color TFT LCD Display.” *Crystalfontz America Incorporated*,

Crystalfontz, 9 Nov. 2017, www.crystalfontz.com/product/cfaf128128b10145t-128x128-graphic-tft-spi.

Diehr, Mark. “PGF - Online.” *PGF - Online*, www.programmingfoundations.com/.

“Digital Game Design I” *Academic Calendar & Catalogs Interactive Media and Game Development*, Worcester Polytechnic Institute, www.wpi.edu/academics/calendar-courses/course-descriptions/interactive-media-game-development#imgd_2900.

“Digital Game Design II” *Academic Calendar & Catalogs Interactive Media and Game Development*, Worcester Polytechnic Institute, www.wpi.edu/academics/calendar-courses/course-descriptions/interactive-media-game-development#imgd_3900.

Feltham, Jamie. “Holoplayer One Is An Interactive Lightfield Display Bringing Holograms To Your Home.” *UploadVR*, UploadVR, 21 Nov. 2017, uploadvr.com/holoplayer-one-interactive-lightfield-display-bringing-holograms-home/.

“Getting Started with the HoloPlayer Three.js Library.” Through the Looking Glass, Medium, 28 Nov. 2017, blog.lookingglassfactory.com/getting-started-with-the-holoplayer-three-js-library-86bdbeca351.
Hara, Minna, and Saila Ovaska. Heuristics for Motion-Based Control in Games, ACM, 2014, doi:10.1145/2639189.2639246.

Hedgecock, Will. “JSerialComm.” Platform-Independent Serial Port Access for Java, 1.3.11, Fazecast, Inc., fazecast.github.io/jSerialComm/.

Higgins, Sean. “HoloPlayer One: Turn your 3D data into an interactive hologram.” Spar3D. 6 December 2016. Web. URL.

Looking Glass. “HoloPlayer One.” Looking Glass, Looking Glass, 2017, lookingglassfactory.com/product/holoplayer-one/.

“Magnetic Liquid Technology.” Ferrofluid, ferrofluid.ferrotec.com/technology/.

"Maya." Maya | Computer Modeling & Animation Software | Autodesk, Autodesk, www.citationmachine.net/mla/cite-a-software/manual.

Melville, Ryan, and Patrick Petersen. "Whiter." Whither, users.wpi.edu/~bmoriarty/ps/examples/whither/cover.html.

Moriarty, Brian J. "Perlenspiel." Perlenspiel3, 3.2, 2009, users.wpi.edu/~bmoriarty/ps/index.html.

Moriarty, Brian J. "API." Perlenspiel3, 3.2, 2009, users.wpi.edu/~bmoriarty/ps/api.html.

Moriarty, Brian J. "Lehr und Kunst mit Perlenspiel." GDC Education Summit. GDC Education Summit, 2 Feb. 2018, San Francisco, GDC Vault, California, www.gdcvault.com/play/1015620/Lehr-und-Kunst-mit.

"Nashorn." Oracle Nashorn: A Next-Generation JavaScript Engine for the JVM, Oracle, 2014, www.oracle.com/technetwork/articles/java/jf14-nashorn-2126515.html.

Norman, Don. The Design of Everyday Things. Basic Books, 2013.

Noessel, Christopher. "What Sci-Fi Tells Interaction Designers About Gestural Interfaces." Smashing Magazine, 1 Mar. 2013, www.smashingmagazine.com/2013/03/sci-fi-interaction-designers-gestural-interfaces/.

"OpenGL." The Industry Standard for High Performance Graphics, Khronos Group, www.opengl.org.

Plemmons, Daniel, and Paul Mandel. "Designing Intuitive Applications." Leap Motion Developers, developer-archive.leapmotion.com/articles/designing-intuitive-applications.

"Reaper." Reaper Digital Audio Workstation, 5.77, Cockos Incorporated, www.reaper.fm.

Roberts, David E. History of Lenticular and Related Autosterioscopic Methods. Leap Technogies, 2003.http://www.microlens.com/pdfs/history_of_lenticular.pdf

"ScriptEngine put()." Java SE, 8, https://docs.oracle.com/javase/8/docs/api/javax/script/ScriptEngine.html#put-java.lang.String-java.lang.Object-.

Spiel Matrix. "Perlenspiel Game Resources & Student Examples." Spielmatrix, 2013, spielmatrix.com.

Sporka, Adam, et al. Non-Speech Input and Speech Recognition for Real-Time Control of Computer Games, vol. 2006, ACM, 2006, doi:10.1145/1168987.1169023.
Timmer, John. "New hardware lets any computer run an interactive, 3D interface." Ars Technica. 21 November 2017. Web. URL.

“URL.” Java SE, 6, docs.oracle.com/javase/6/docs/api/java/net/URL.html.

“URL openStream().” *Java SE*, 6, https://docs.oracle.com/javase/6/docs/api/java/net/URL.html#openStream%28%29.

“What Is Holography? | Holocenter.” *Center for the Holographic Arts*, Holocenter, holocenter.org/what-is-holography.

Wong, Kevin. “The Science Behind The World's Most Convincing Ghost Effect.” *Motherboard*, VICE, 28 Oct. 2016, motherboard.vice.com/en_us/article/aekye5/the-science-behind-the-worlds-most-convincing-ghost-effect.

“ZBrush.” *ZBrush | The All-in-One-Digital Sculpting Solution*, Pixologic, pixologic.com.