Smart Mirror: Final Report

Abstract

In our daily routines, we often visit multiple applications. One enthusiast solution to combine these applications is the smart mirror -- a mirror that serves a reflection and displays information. There exists a community that develops features for smart mirrors -- for example: to-do lists, stock viewers, and even video browsers. We saw room to solve two problems. First, we saw that consistently recording 1-second-a-day videos can be difficult. Second, we saw that gamers wanted to stay updated on changes to game. Thus, we decided to leverage this platform to add two modules -- a 1-second-a-day module and a game patch notes module. Respectively, these would make it easier for people to build their own 1-second-a-day videos and stay current on game updates.

Introduction

A smart mirror is a display that functions as both a reflective mirror and a computer display. Currently, there is an existing community of smart mirror enthusiasts who both build smart mirrors and develop software for them. MagicMirror^2 is a smart mirror platform that offers basic functionality like time, weather, and calendars, but also is extensible with "modules". Any developer can build a "module" for MagicMirror^2 that can add extra functionality to a smart mirror. For example, there are modules for to-do lists, sports, YouTube, travel, traffic tracking, and more. In total, there are exists an ecosystem of over 700 modules.

In this project, we accomplished three main things. First, like many others in the community, we built our own smart mirror, learning the necessary skills along the way. Second, we developed a novel 1-second-a-day module, which takes the idea of 1-second-a-day-videos and makes them easier to create with a smart mirror. Third, we developed a patch notes module, which displays information on the latest updates on games (currently, Overwatch).

For several years, many people have created "1-second-a-day" videos. A 1-second-a-day video displays 1 second of each day of someone's life over period of time and serves as a timelapse of one's life. These videos offer a fun way to share your life with others. However, 1-second-a-day videos are often only made by professional content creators, as it can be difficult to consistently dedicate time to building a several-years-long project.

Given that checking your mirror is a daily routine to any smart mirror user, we saw room for a new idea -- a smart mirror 1-second-a-day module. With this module, we saw that we could:

- take advantage of the external webcams found on most smart mirrors
- provide a convenient platform from which someone could easily record and compile 1-second-a-day clips
- keep users consistent with creating videos by serving reminders in a place that they always check on a daily basis

In particular, we:

- Built a voice-controlled module from which users could record, compile, and upload 1-second-a-day videos
- Built a user interface that would respond to these commands
- Built a user interface that displayed statistics to keep users accountable

Second, we saw room to implement another module, the patch notes module. Active gamers want to stay informed on the latest updates to their game, so we built a module that displays patch notes on the smart mirror.

We have since open-sourced and published both of these modules on MagicMirror^2 public list of modules.

Technology

Our project can be broken down into three major parts:

1. Smart Mirror construction
2. 1-second-a-day module development
3. Patch notes module development

Smart Mirror Construction

Our smart mirror consists of five main components:

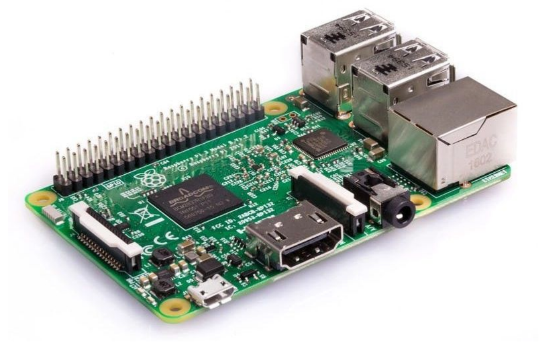
1. A Monitor
2. A Computer
3. A Webcam
4. A Two-way mirror
5. A Frame

Before building our mirror, we had to give careful attention in researching and choosing the parts. Many options for components can conflict with other components due to physical or software constraints.

<u>Monitor: HP 24uh</u>

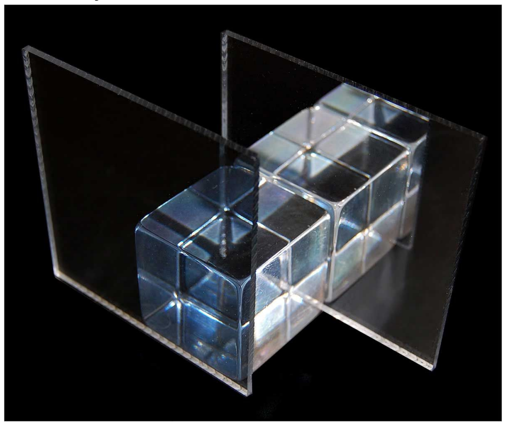
We chose the HP 24uh as our monitor for multiple reasons. First, the monitor was 24" wide diagonally. This was large enough for a person to see most of their upper body in their reflection but small enough so our two-way mirror didn't have to be sized at an inordinately expensive size. Second, the bezel of the monitor was removable via a flathead screwdriver. This allows us to arrange the monitor's display flush against the two-way mirror, resulting in a sharper image.

<u>Computer: Raspberry Pi 3</u>



The Raspberry Pi is a compact $35 computer popular in many home internet-of-things projects. We chose the Raspberry Pi because of its small size and low cost.

Two-way Mirror:



A smart mirror's mirror can be made of reflective film, glass, or acrylic. We decided to go with a ready-cut two-way mirrored acrylic from TapPlastics. Acrylic is superior to film, in that it allows more light from the monitor to shine through. At the same time, it is cheaper and less fragile than glass.

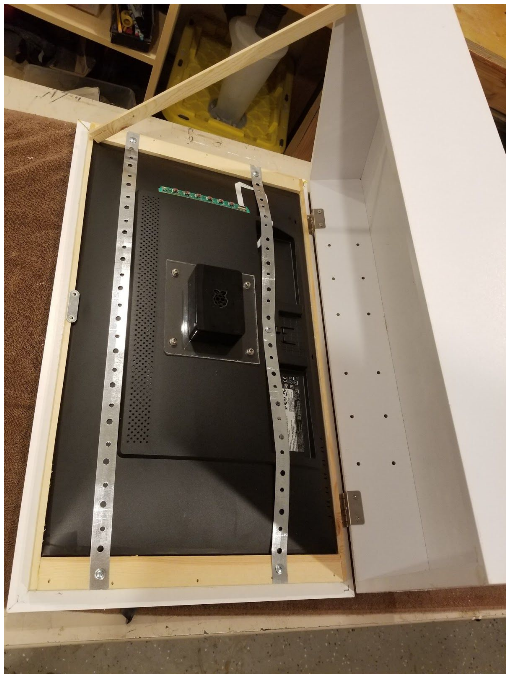
Frame: Glacier Bay 15 1/4in x 26in Medicine Cabinet

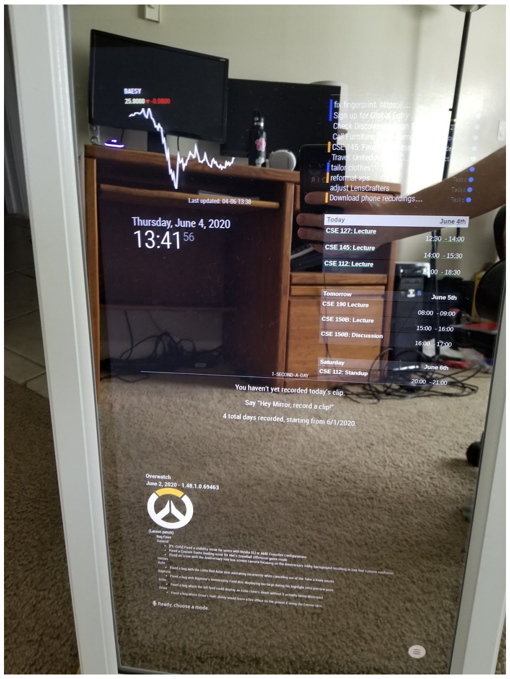
We chose this medicine cabinet as our frame. First, its mirror opening is the perfect size -- only slightly larger than our HP 24uh monitor. Second, its hollow body provides just enough room to store the monitor. Third, its built-in hinge allows us to easily open up the smart mirror and access/adjust its internal components.

Frame Build Process

We started by removing the bezel on the monitor and replacing it with black electrical tape that ensured that we couldn't see the outline of the monitor through the glass. We then pulled out the little circuit board for the buttons to adjust brightness, turn monitor on, etc. We taped this to the back of the monitor for occasional future use if necessary. Next we broke the old glass that was included in the frame housing and replaced it with the new two way glass that we purchased.

We then cut out several pieces of wood that were used to surround the monitor and keep it in place. After doing so, we glued these pieces to the frame housing/monitor and put it in place. Next we painted the outside of the frame to clean up any blemishes done in the process. We grabbed some bolts and placed them on top and bottom of the wood pieces and put flexible metal strips running across the back of the monitor to ensure it doesn't move too much. Lastly we cut a piece of plexiglass and screwed it onto the back of the monitor and screwed our raspberry pi into it. After hooking all the cables, we completed the construction of the monitor.


(In-progress picture without cables)



(Final product)

MagicMirror2 Platform Background

MagicMirror^2 is an existing open source platform for smart mirrors. It provides the software infrastructure for an always-on smart mirror and comes prepacked with basic functionality, such as the time and weather. In addition, it provides a foundation for other developers to build, connect, and distribute their own modules with new functionality.

MagicMirror^2 and its modules are developed using HTML, CSS, and JavaScript. ElectronJS is used as a desktop application wrapper on top of these languages. In its backend, MagicMirror^2 and its modules are built with NodeJS.

1-Second-A-Day Module Development

At a high level, our 1-second-a-day module

1. Input a voice command to record a 1-second clip.
2. Input a voice command to compile a clip.

Voice Control

If the webcam hears a user say "Magic Mirror, Create Clip", the smart mirror will record a 1-second clip. When the user says "Magic Mirror, Create Compilation", the smart mirror will compile these clips into one video and upload the video to Google Drive.

We used the PocketSphinx speech recognition engine for our 1-second-a-day module. Though more advanced engines, such as Google's Speech API, exist, we chose PocketSphinx for two reasons. First, it is sophisticated enough to accurately recognize our two commands. Second, it is lightweight enough to run on the Raspberry Pi, thus reducing the number of external dependencies and API keys a user of our module would need to have.

Video Recording

1-second clips were recorded from the the smart mirror's webcam using the MediaDevices.getUserMedia() API. This allowed us to capture input from the user's webcam, display it back to the user, and save it locally on the raspberry pi.

Video Compilation

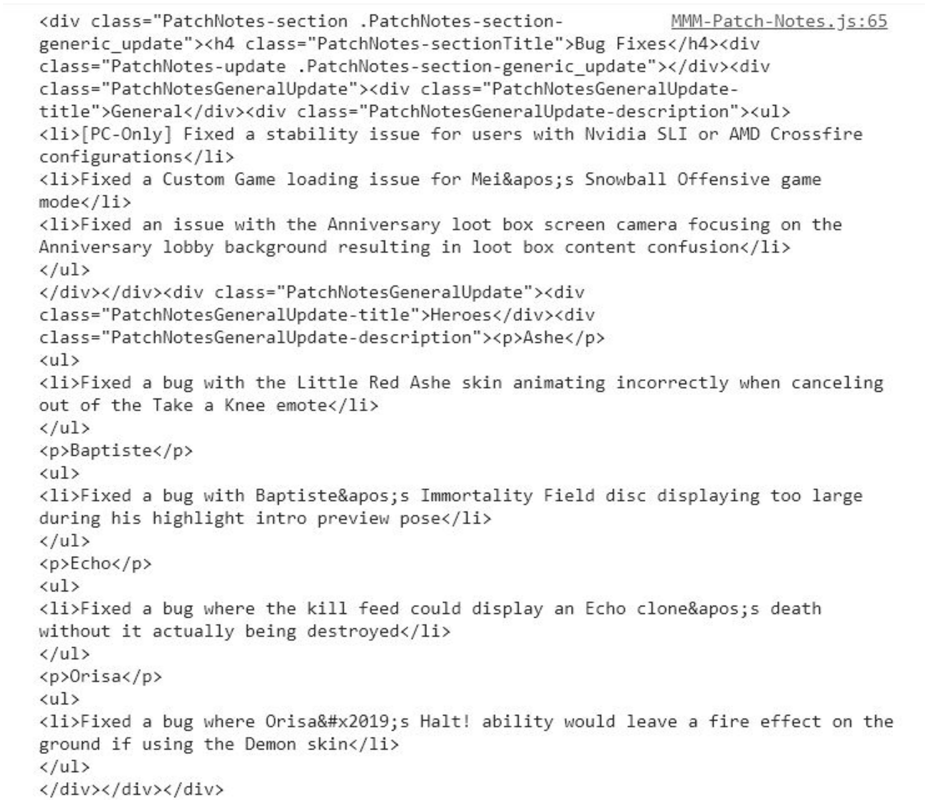
1-second clips were compiled into a longer video using the Fluent-FFMPEG library.

Google Drive Uploading

Since accessing the storage of a smart mirror's Raspberry Pi is often inconvenient, we uploaded compilations to the cloud using Google Drive's API. This requires users to configure OAuth authentication with our module.

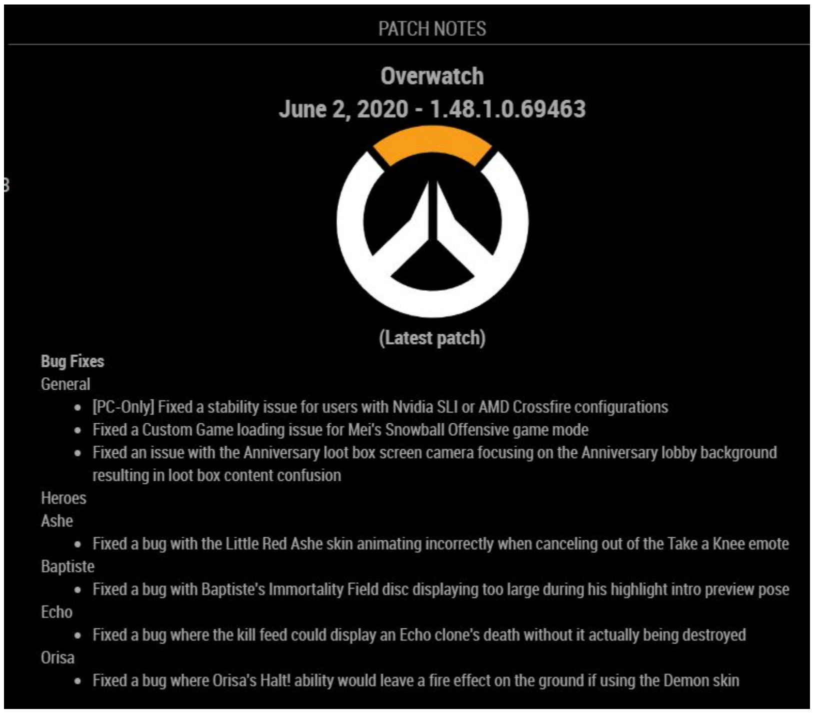
Patch Notes Module Development

Our patch notes module uses Axios to form a HTTP request to playoverwatch.com. We then scrape the response's HTML to receive output similar to the following:



```
<div class="PatchNotes-section .PatchNotes-section-          MMM-Patch-Notes.js:65
generic_update"><h4 class="PatchNotes-sectionTitle">Bug Fixes</h4><div
class="PatchNotes-update .PatchNotes-section-generic_update"></div><div
class="PatchNotesGeneralUpdate"><div class="PatchNotesGeneralUpdate-
title">General</div><div class="PatchNotesGeneralUpdate-description"><ul>
<li>[PC-Only] Fixed a stability issue for users with Nvidia SLI or AMD Crossfire
configurations</li>
<li>Fixed a Custom Game loading issue for Mei&apos;s Snowball Offensive game
mode</li>
<li>Fixed an issue with the Anniversary loot box screen camera focusing on the
Anniversary lobby background resulting in loot box content confusion</li>
</ul>
</div></div><div class="PatchNotesGeneralUpdate"><div
class="PatchNotesGeneralUpdate-title">Heroes</div><div
class="PatchNotesGeneralUpdate-description"><p>Ashe</p>
<ul>
<li>Fixed a bug with the Little Red Ashe skin animating incorrectly when canceling
out of the Take a Knee emote</li>
</ul>
<p>Baptiste</p>
<ul>
<li>Fixed a bug with Baptiste&apos;s Immortality Field disc displaying too large
during his highlight intro preview pose</li>
</ul>
<p>Echo</p>
<ul>
<li>Fixed a bug where the kill feed could display an Echo clone&apos;s death
without it actually being destroyed</li>
</ul>
<p>Orisa</p>
<ul>
<li>Fixed a bug where Orisa&#x2019;s Halt! ability would leave a fire effect on the
ground if using the Demon skin</li>
</ul>
</div></div></div>
```

From this, we generate and display our own UI to relay this information.



PATCH NOTES

Overwatch

June 2, 2020 - 1.48.1.0.69463

(Latest patch)

Bug Fixes

General

- [PC-Only] Fixed a stability issue for users with Nvidia SLI or AMD Crossfire configurations
- Fixed a Custom Game loading issue for Mei's Snowball Offensive game mode
- Fixed an issue with the Anniversary loot box screen camera focusing on the Anniversary lobby background resulting in loot box content confusion

Heroes

Ashe

- Fixed a bug with the Little Red Ashe skin animating incorrectly when canceling out of the Take a Knee emote

Baptiste

- Fixed a bug with Baptiste's Immortality Field disc displaying too large during his highlight intro preview pose

Echo

- Fixed a bug where the kill feed could display an Echo clone's death without it actually being destroyed

Orisa

- Fixed a bug where Orisa's Halt! ability would leave a fire effect on the ground if using the Demon skin

Project Milestones

Our project was composed of five high-level deliverables, each in turn composed of multiple smaller milestones. Understanding our lack of familiarity with the involved technologies/skills, we started the quarter with a smaller number of milestones. In the middle of the quarter, we were more comfortable with development and saw room to add more milestones. Specifically, we did two things. First, we increased the complexity of our 1-second-a-day module, adding voice control and a UI. Second, we decided to build to also build a "Patch Notes" module, which displays information on the latest updates for games.

Priority Key

Priority	Description
P0	Critical
P1	Important
P2	Stretch goal

Deliverable: Smart Mirror Construction (3 weeks)
Acceptance Criteria: The mirror is assembled into the frame in one piece. An image is produced where it is possible to see both the reflection of the user and the display (when turned on) behind them.

Milestone	Time Estimate	Priority
Research and purchase materials	1 week	P0
Build frame and attach 2-way glass to it	1 week	P0
Add supports for display, and mount display inside frame	1 week	P0

Deliverable: Initial Setup (1 week)
Acceptance Criteria: An image is produced where we can see the magic mirror software running on the raspberry pi with the existing modules on screen.

Milestone	Time Estimate	Priority
Set up Raspberry Pi 3	0.3 weeks	P0
Install and test Magic Mirror 2 software	0.3 weeks	P0
Incorporate existing modules - time, calendar, to-do list	0.3 weeks	P0

Deliverable: Build 1-second-a-day Module (4.5 weeks)
Acceptance Criteria: When the user says "Magic Mirror, Create Clip", a 1-second-long video is taken and stored locally. When the user says "Magic Mirror, Create Compilation", all previous 1-second videos are compiled into a video compilation and uploaded to a folder in the user's Google Drive.

Milestone	Time Estimate	Priority
Implement voice control trigger	1 week	P2
Implement video recording	1 week	P1
Implement video compilation	1 week	P1
Implement uploading recordings to Google Drive	.5 week	P2
Implement 1-second-a-day UI	1 week	P2

Deliverable: Build Patch Notes Module (4.5 weeks)
Acceptance Criteria: When loaded, the smart mirror displays the latest patch notes from the game Overwatch.

Milestone	Time Estimate	Priority
Scraping	1 week	P2
UI	1 week	P2

Deliverable: Course Tasks
Acceptance Criteria: All course tasks, documents, and presentations are submitted to the professor and/or presented to the class.

Milestone	Time Estimate	Priority
Oral Project Update	1 week	P0
Project Web Presence	1 week	P0
Final Oral Presentation	1 week	P0
Final Project Video	1 week	P0
Final Report	1 week	P0

Major Challenges

Shipping Delays

Due to the COVID-19 pandemic, we faced shipping delays for our monitor and two-way acrylic mirror. This delayed the construction of our mirror by several weeks. To counteract this, we adjusted our milestones schedule, pushing back the mirror's construction while bringing closer software milestones.

Remote Challenges

The remote circumstances of the project resulted in three challenges. First, it forced us to reallocate roles. We had initially planned on splitting the work for both the hardware and software aspects of the project, as we were both curious to learn about both parts. However, given that we could not meet physically, we had to allocate the mirror construction aspect of the project entirely to Kyle. Second, it made testing code on the mirror difficult. Since Gary did not have physical access to the mirror, when implementing voice control, we had to go back and forth in attempting to debug issues with voice control in the smart mirror.

Google Drive Authentication

To enable users of our module to upload to their Google Drive we have to authenticate with their accounts. There were a lot of issues getting this to be a smooth experience for users, as most implementations require the user to perform several steps. We ended up implementing it in a way we saw as smooth as possible, while still not being ideal. In addition, there were several problems where Drive wasn't authenticating with certain accounts, etc. These were mostly solved, but without many users testing is not confirmed if it's completely solved.

Conclusion

In conclusion, our project delivered in accomplishing our three major goals. First, we planned and executed building a working smart mirror from scratch. Second, we built a open source 1-second-a-day module, allowing anybody with a Smart Mirror to record their own 1-second-a-day videos. Third, we built a patch notes module, allowing Overwatch gamers to stay updated on the game's latest patches. We've expanded the community's system of modules and have offered our own take and documentation on its development.

With each goal, we dived into new territories. Kyle learned about the process behind constructing a physical mirror: how to cut, sand, a paint wood. Gary learned about the architecture behind the Magic Mirror platform, and how he can contribute to growing an open source community.

Moving forward, both of us intend to continue this project. Kyle intends to continue developing the patch notes module, expanding to games like CS: GO and Valorant and adding features like scrolling by voice command. Gary intends to build his own larger and more feature-rich smart mirror, diving into the potential of a touch-screen mirror or a gesture-detecting mirror using a Leap Motion.

References

MagicMirror^2