CySyphus Kinetic Art Table

Team Members: Sean Gordon, Morgan Funk, Aaron Lawrence, Daniel Laracuenta Cortes, William Maston, and Samuel Christianson

SDDEC20-04

Client and Faculty Adviser: Doug Jacobson

Project Overview

The Sisyphus kinetic art table lacks an interface for creating new tracks to be displayed. Users have to manually make polar coordinate files to interact with the table, and then upload them to a Raspberry Pi.
We decided to make it so users can upload images that would automatically converted into table drawing files. These files would also be submitted via a website.

Users and Uses

This project was built to allow for any novance user that had access to the controlling website to create a account and upload any image of their choosing.

Frontend Website

Testing

 React: A programming plugin for JS website design. Anytime we made a change to our code base, the changes would be immediately displayed in real time. This allowed for fast and easy debugging and testing of features.

Functional Requirements

- Be able to convert image into polar coordinate files
- Non-Functional Requirements
- Have a companion website to go with the table
- Have the website be able to support image submission and create Playlists of images to draw
 Run a twitch stream to watch the table remotely

CySyphus Kinetic Art Table Who are we? My Table Sign UP Log In **Community Tracks** 🗙 Name Sam's Table Queue Download Currently Playing By CySyphus [0] PREVIEW o preview availa Track3 REVIEW Pause 🕨 Skip 0 1 0 1 0 1 Upload New Track React upload files AndLess 01 Choose File No file cl Blonded 1 ۵ 🖬 Backend Conceptual

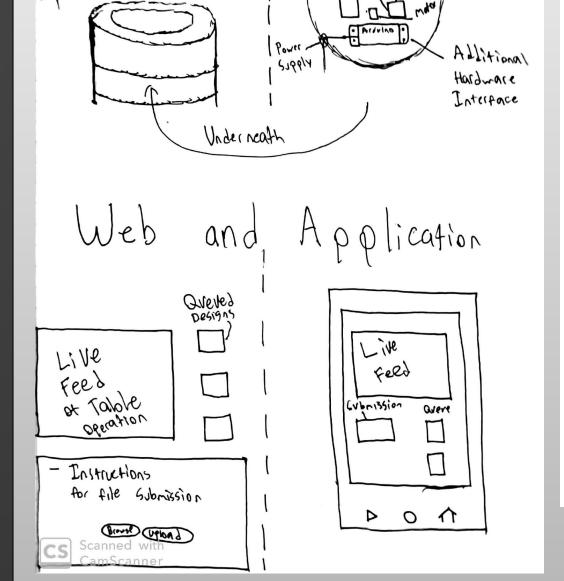
• When sending data to the table, we utilized postman requests to carry the data to the table. Later on we automated these requests to an extent using python to further improve on the image drawing algorithm that converted cartesian points to polar points. This was a tedious process due to the many rules the table itself had in place for drawing.

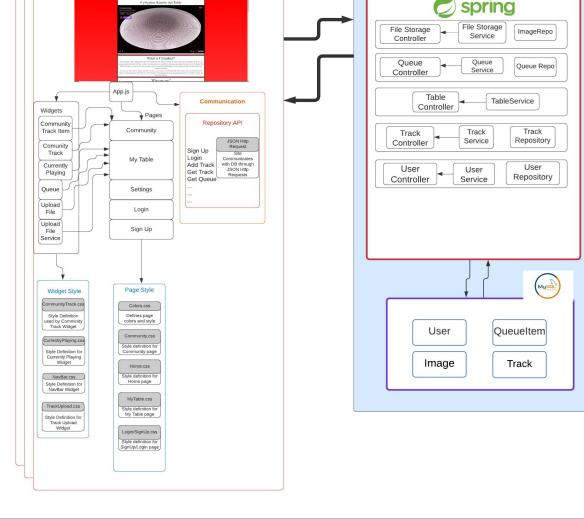
Desig	gn Approach
oncept Sketch	Block Diagram
Rasberry motor Fill Controller	Aga Block Diagram Morgan Funk November 15, 2020

Operating Environment

Physical art table using a Raspberry Pi
A webserver using Java and SQL
A website running off of Javascript

QueueController.java	34 public class Fliecontroller {
> 🚺 TableController.java	35
> 🚺 TrackController.java	<pre>36 private static final Logger = LoggerFactory.getLogger(FileController.class); 37</pre>
> 🚺 UserController.java	sr Sr∰ Biutowired
🗸 🖽 exception	39 private FileStorageService fileStorageService;
> 🚺 FileStorageException.java	40
> 🌆 MyFileNotFoundException.java	41● @Autowired
🗸 🧱 > model	42 private TrackService trackService; 43
🔉 👬 > table	A BPostHapping("/uploadFile")
> 🋂 Image.java	45 public UploadFileResponse uploadFile(@RequestParam int userId, @RequestParam("file") MultipartFile file) {
> 🎝 User.java	46 // if(!userService.exists(name)) return new UploadFileResponse("This user does not exist", "", "", 0);
🗸 👪 payload	47
> 🛂 PauseResponse.java	<pre>48 String fileName = fileStorageService.storeFile(file); 49</pre>
> 🚺 PlayResponse.java	50 String fileDownloadUri = ServletUriComponentsBuilder.fromCurrentContextPath()
> 🚺 QueueltemResponse.java	51 .path(*/downloadFile/*)
> 🌠 TimeResponse.java	52 .path(fileName)
> 🚺 UploadFileResponse.java	53 .toUriString();
🛩 👪 property	<pre>54 55 List<string> directories = fileStorageService.convertToTrack(fileName);</string></pre>
> 🋂 FileStorageProperties.java	56
🗸 🏭 > repository	57 trackService.addTrack(directories, fileName, userId);
> 🋂 ImageRepository.java	58 Last statut last last last last last last last las
> 🋂 PlaylistRepository.java	59 return new UploadFileResponse(fileName, fileDownloadUri,
> 🋂 QueueRepository.java	<pre>60 file.getContentType(), file.getSize()); 61 }</pre>
> 🛃 StreamRepository.java	62
> 🛂 TrackRepository.java	630 @PostMapping("/uploadMultipleFiles")
> 🚺 UserRepository.java	64 public List <uploadfileresponse> uploadMultipleFiles(@RequestParam int userId, @RequestParam("files") MultipartFile[] files) {</uploadfileresponse>
✓ dia > service	65 // if(luserService.exists(name)) return new ArrayList <uploadfileresponse>(Arrays.asList(new UploadFileResponse("This user does not exist", "", "", 0)));</uploadfileresponse>
> 🚺 FileStorageService.java	66 return Arrays.asList(files) 67 .stream()
> 🛺 > QueueService.java	68 .map(file -> uploadFile(userId, file))
> 🚺 TableService.java	69 .ccllet(Collectors.tolist());
> 🚺 > TrackService.java	78 }
> 🚺 UserService.java	71
> 🚺 WebServiceApplication.java 🔍 🗸	720 @GetMapping("/downloadFile/{fileName:.+}") 73 public ResponseEntity <resource> downloadFile(@PathVariable String fileName, HttpServletRequest request) {</resource>
	73 public Response intry Resource 2 dominatorial (gradinor date string filename, https://iteau.com/
	75 Resource = fileStorageService.loadFileAsResource(fileName);
t Repositories 🗙 📃 🗖 🔳	



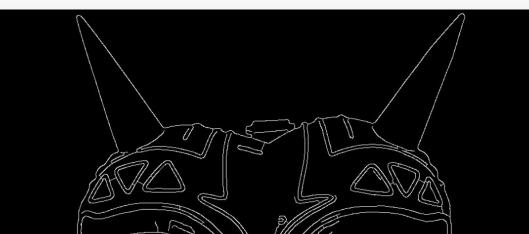


Technical Details

The table itself has a raspberry pi
 that does the bulk of the data
 processing and drawing. This pi
 handles the lights, motors and
 communication between a table









app and any postman request we send it. It also facilitates the drawing of images though theta rho files (.thr) These files are the key to the tables drawing and designs created.

