CDM

CENTRE FOR DIGITAL MEDIA | 685 GREAT NORTHERN WAY | VANCOUVER, BC CANADA | V5T 0C6 | theCDM.ca

Sample of Work Example:

JongWon Kim

Cohort 12

South Korea

Programmer

Method of Submission:

This applicant uploaded documents directly into the SFU Graduate Admission Application Portal.

Location of Materials:

See Sample of Work Summary and documents below.

Sample of Work Summary

JongWon(Karl) Kim

Empowered by digital technologies, media is changing relationships between people and cities. For example, *Uber* changes the role of citizens from passive consumers of municipal services into active contributors. This allows people to positively influence their cities' transportation system by filling gaps between private and public transportation. While studying at Georgia Institute of Technology, I have researched and created digital mediums that can reshape these interactions between citizens and cities.

Through the *FoodParent* project, I designed and programed both the server-side and the client-side of the application by utilizing modern web technologies, *HTML5 canvas rendering, Ajax,* and *Flux architecture*. Via this responsive web application, people can easily participate the *FoodParent* program to increase food donations for supporting more homeless people in Atlanta.

Drones for Foraging is a project that I had participated in as a software engineer, and which also received notable mentions in the *Core77 Design Awards* student speculative category in 2014. The idea of the project was using relatively cheap flying drones to track fruit ripening of trees, along with crowdsourcing methods, like *FoodParent*. I have developed an *Android* application that communicates with a quadrocopter to navigate to and detect fruit ripening of the trees autonomously by utilizing socket network, GPS tracking, real-time video (de)compression, image processing, and computer vision.

The Kiln is a project requested by Dr. Russell Gentry, Associate Professor of Architecture and Civil Engineering at Georgia Institute of Technology, to provide a tool that helps architects choose the most suitable brick textures for their works. Performing as a lead software engineer, I developed a *Python* server that communicated with a 3D computer graphic rendering tool, *Blender*, to generate photo-realistic brick textures based on the user's selections of parameters, such as colors, dragging marks, and cracks on a brick surface.

Increasing people's understanding of smart cities is as equally important as providing great tools to citizens. To improve smart city literacy of citizens in Atlanta, I've built an *Arduino*-based sensor box with an industrial design student at Georgia Tech, Natalie Larkins, that allows participants of the smart city workshop to bring the box outside, observe data generated from it, and use the data to understand about benefits of smart city.

As I believe future cities need to become more playful places to people, I have studied game design at Georgia Tech, and have developed computer and augmented reality games using various platforms, including *JavaScript*, *Unity*, and *Unreal Engine*. *Escape Goat 2 (GBA)*, as seen on my portfolio, was developed solely with C language. I built a physics engine to handle collision detection between the character and platforms, an asset manager to handle sprite animations, and a custom map (level) editor.

As my portfolio demonstrates, I have various technical skills and knowledge in computer science, as well as a strong desire to create digital mediums that enhance relationships between people and cities, and, eventually, between people. I also firmly believe my skills and desire will be useful in any future digital media projects in the MDM program.

Project 1

FoodParent

PHP JS WEBPACK NODE.JS

Jong Won (Karl) Kim

PHP JS WEBPACK NODE.JS

project_overview.txt

Food Parent is a web-based application as a part of Concrete Jungle's *Food Parent* project. The application provides a tool for volunteers of Concrete Jungle to register new trees on the map, virtually adopt trees and monitor fruit ripening of the trees using mobile devices so that Concrete Jungle can harvest more fruits from untended urban trees and donate them to the needy.

project_clients.txt

- Craig Durkin (Concrete Jungle)
- Katherine Kennedy (Concrete Jungle)

project_members.txt

- Carl DiSalvo (Project Manager, Associate professor, School of Literature, Media, and Communication, Georgia Institute of Technology),
- Jong Won (Karl) Kim (Softeware Developer)
- Jun Ha Park (UI/UX Designer)

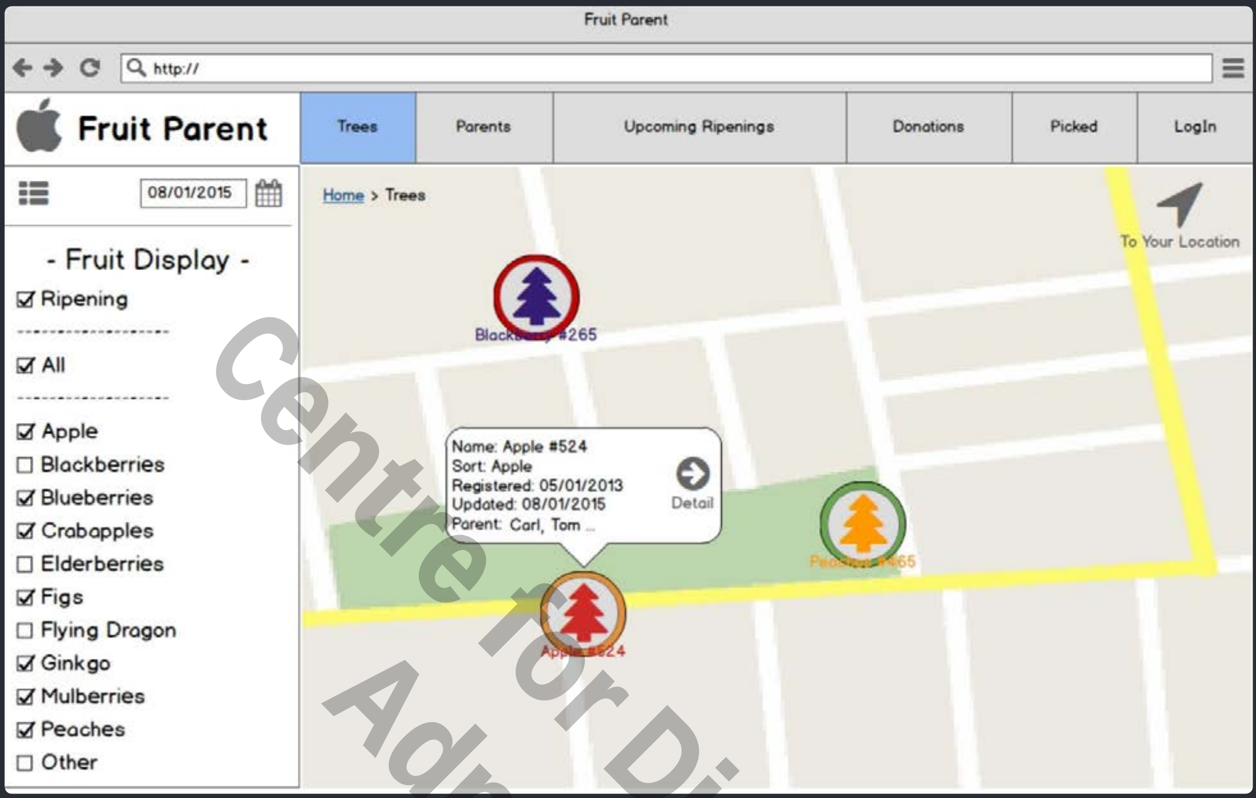
my_contributions.txt

- Built a Geographic Information System (GIS) mapping platform using *Leaflet*.
- Created a custom Canvas-based marker library to render a large number of *Leaflet* markers without performance compromise.
- Built a Single Page Application (SPA) using *React* framework and Flux architecture (*Alt.js*).
- Used ECMAScript 6 syntax with *Babel*'s polyfills.
- Built a server-side API using PHP and MySQL.
- Setup an automated e-mail notification system using *Mailgun*.
- Setup a collaborative coding environment using *Github* and *npm*.
- Setup debugging and bundling configurations using *webpack*.

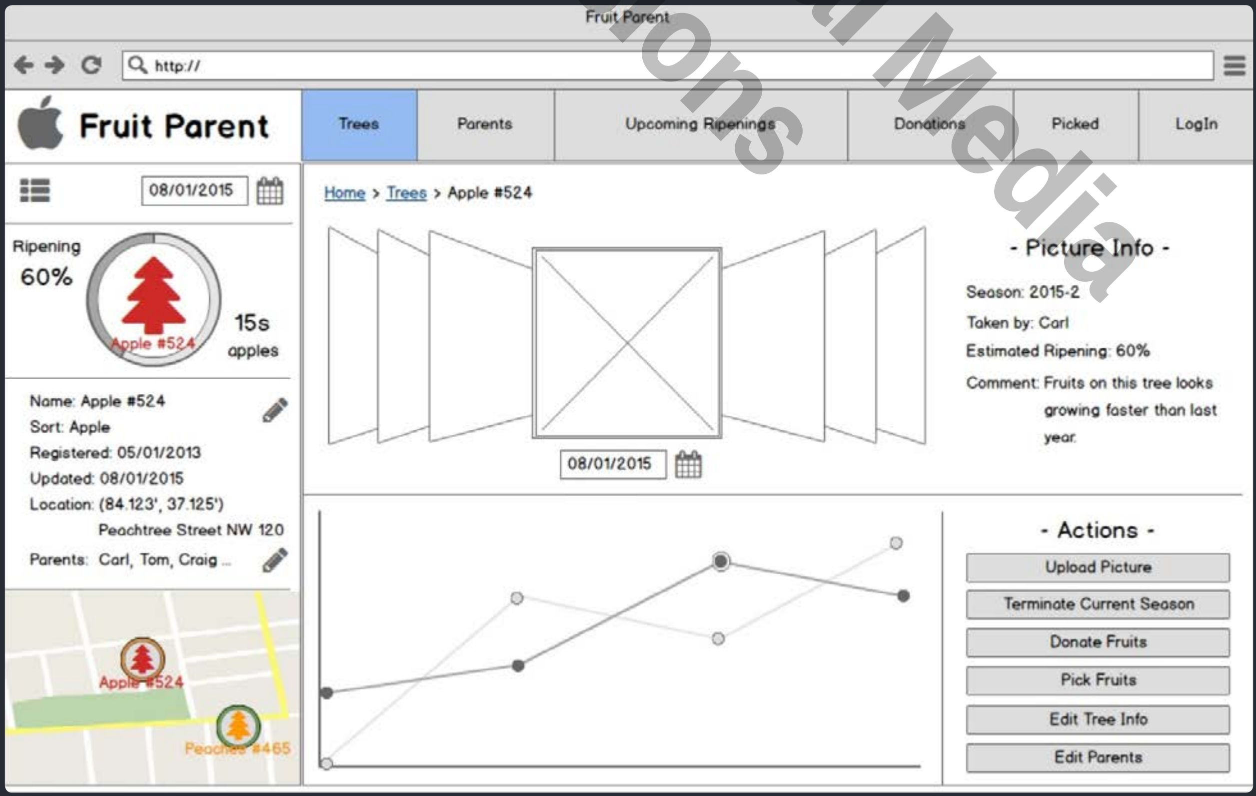
project_outcomes.html

- *Concrete Jungle FoodMap - https://www.concrete-jungle.org/new-fruit-tree-map (it only shows on-season trees by default, so you might need to change the filter on the right panel.)*
- *FoodParent Github Repository - https://github.com/PublicDesignWorkshop/FoodParent2.0*

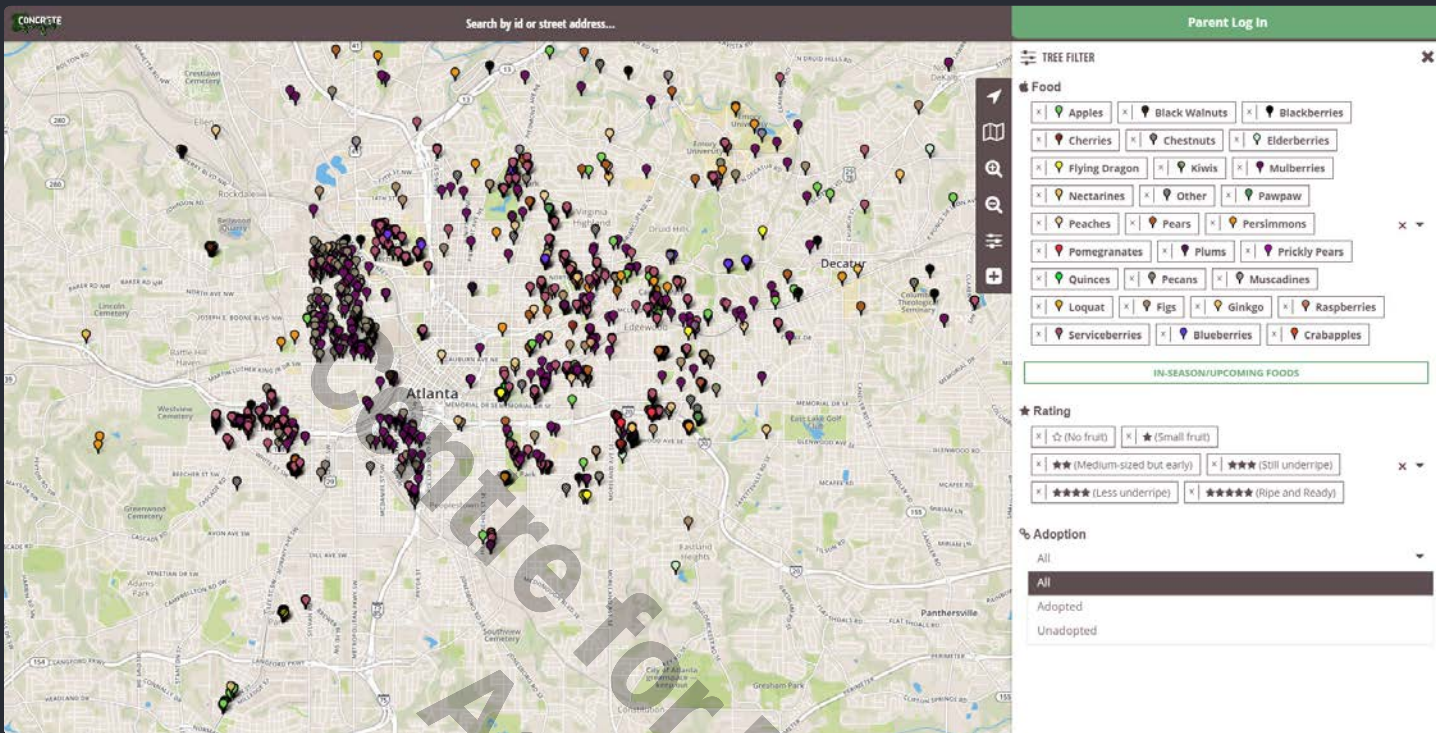
prototyping_balsamiq_1.jpg



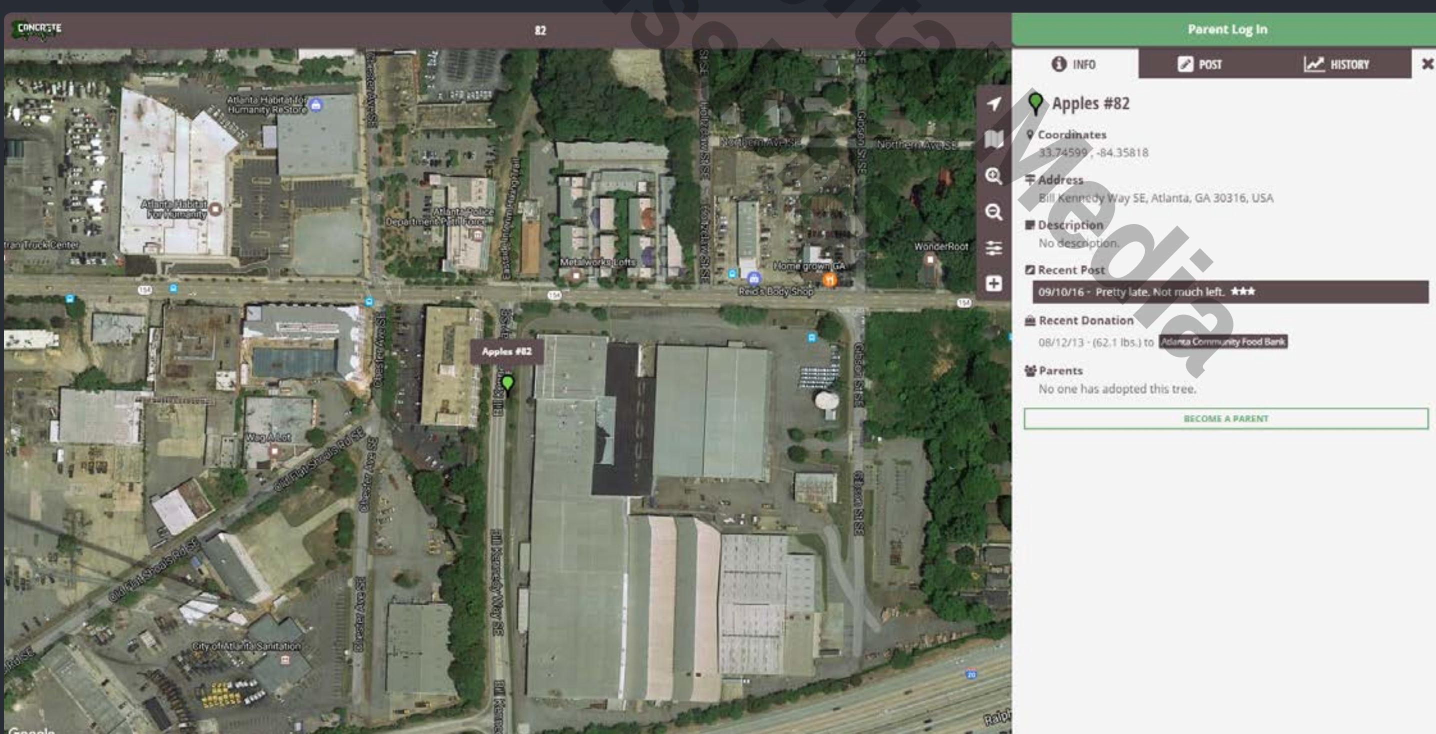
prototyping_balsamiq_2.jpg



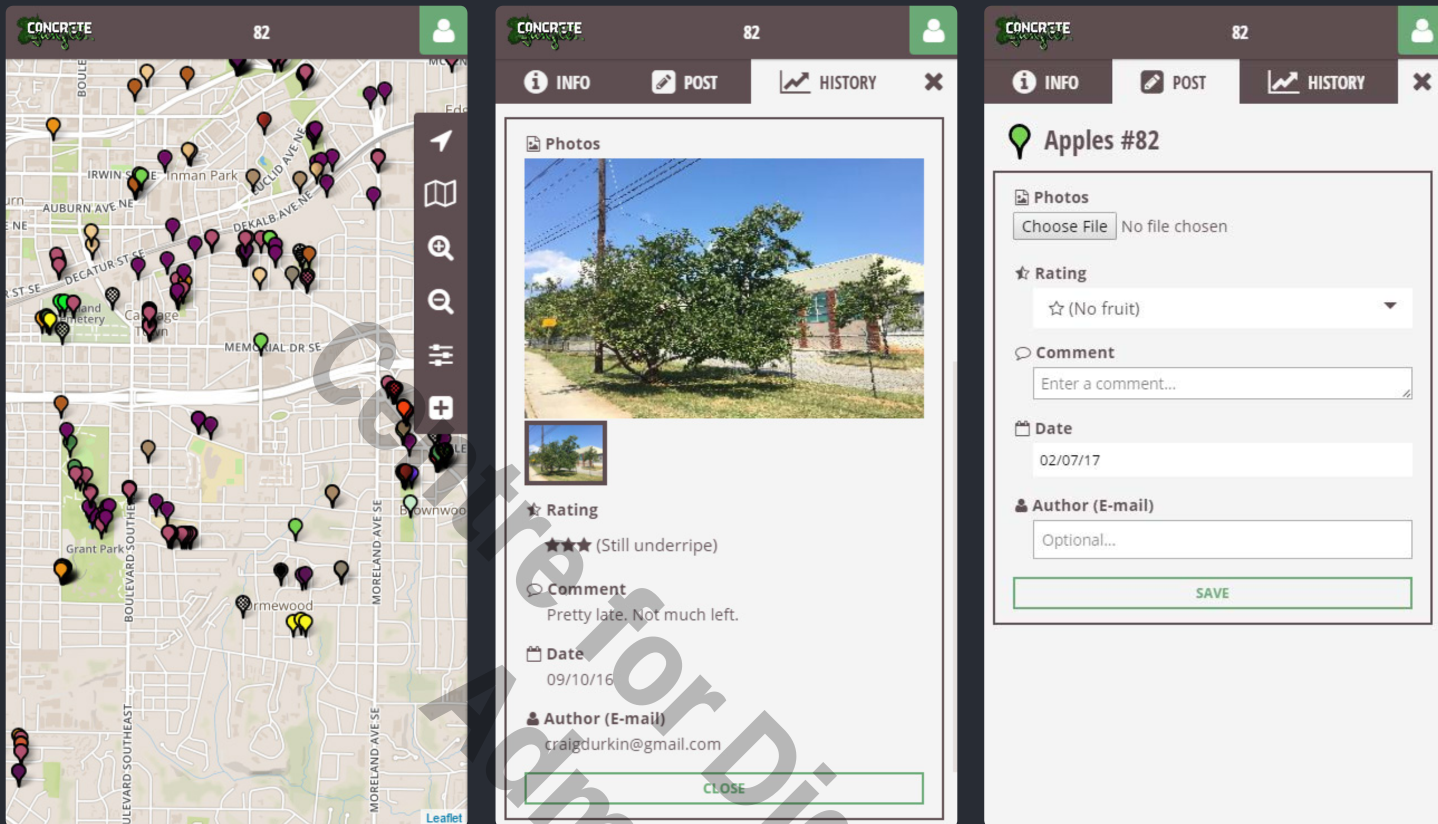
trees_filter.jpg



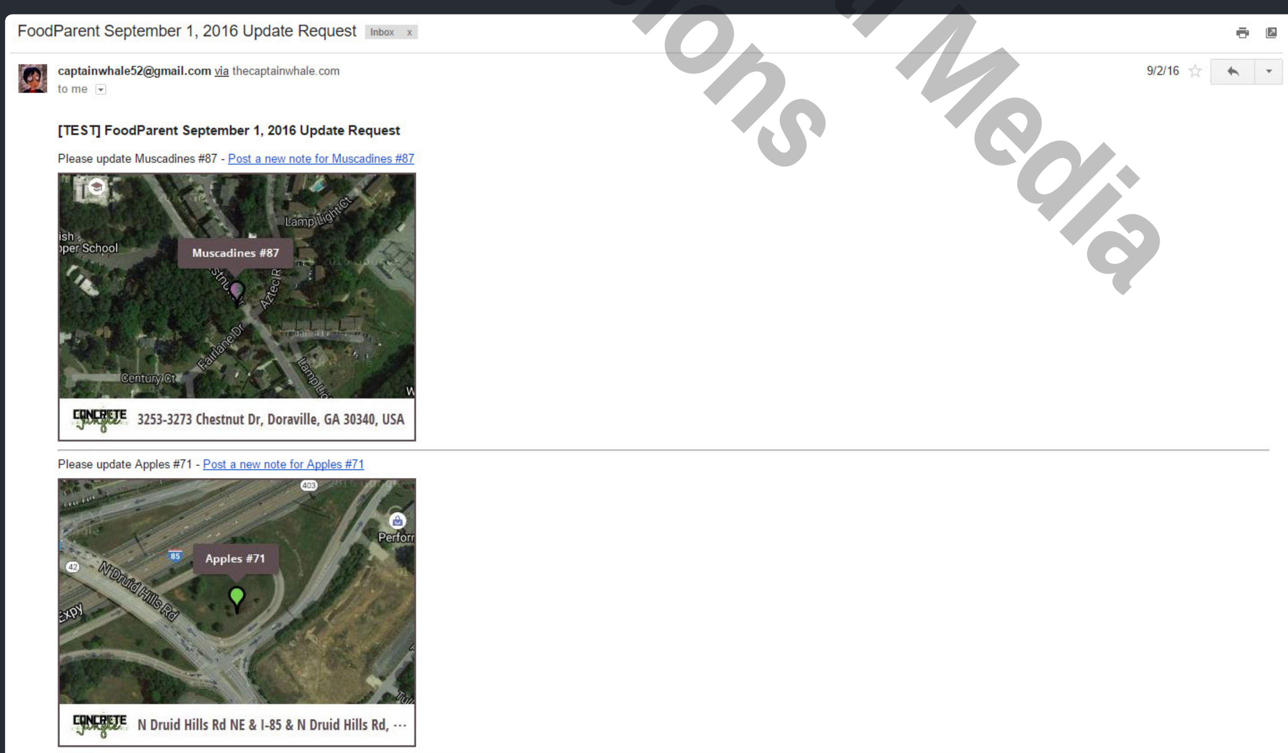
tree_overview.jpg



mobile_layouts.jpg



email_notification.jpg



map-tree.component.js

```
/**
  @file: map-tree.component.js
  @purpose: React component to inject Leaflet GIS mappping llibrary into the React life
cycle, and render markers dynamically depends on filter parameters on the UI.
  @author Karl Kim (captainwhale52@gmail.com)
*/

import React from 'react';
import * as L from 'leaflet';
import 'leaflet-canvas-marker';      // For using a canvas layer rendering pipleline.
import 'googletile';                 // For rendering google map tile.

require('./maps.component.scss');    // Import the scss file for styling the component.

// Import settings from json files.
let MapSetting = require('./../../setting/map.json');
let ServerSetting = require('./../../setting/server.json');

// Import flux architecture components (Alt.js).
let TreeActions = require('./../actions/tree.actions');
let TreeStore = require('./../stores/tree.store');
...   // Import other flux store and action instances.

// Import custom leaflet marker library for rendering canvas markers.
import { createFocusMarker, createCanvasTreeMarker, createSVGTreeMarker }
    from './../utils/marker.factory';

// Define map class using the ES2015 syntax.
export default class MapTree extends React.Component {
  constructor(props, context) {
    super(props, context);
    this.afterRenderMap = this.afterRenderMap.bind(this);
  }

  // Only executed once when the component is created.
  componentDidMount () {
    // Intialize a leaflet instance only once in this class life cyle.
    if (!MapStore.isMapExist( MapSetting.sMapId)) {
      // Initialize a leaflet using the setting values imported from json files.
      this.map = L.map(MapSetting.sMapId, {
          zoomControl: MapSetting.bZoomControl,
          closePopupOnClick: MapSetting.bClosePopupOnClick,
          doubleClickZoom: MapSetting.bDoubleClickZoom,
          touchZoom: MapSetting.bTouchZoom,
          zoomAnimation: MapSetting.bZoomAnimation,
          markerZoomAnimation: MapSetting.bMarkerZoomAnimation,
          minZoom: MapSetting.iMinZoom,
          maxZoom: MapSetting.iMaxZoom,
      }).setView(new L.LatLng(MapSetting.vPosition.x, MapSetting.vPosition.y),
           MapSetting.iDefaultZoom);
      this.map.invalidateSize(false);
      this.map.whenReady(this.afterRenderMap);
```

```
    } else {
      // Import the leaflet instance from the flux store.
      MapActions.setMapType(MAPTYPE.TREE);
      this.map = MapStore.getMapModel(MapSetting.sMapId).map;
      this.flatTileLayer = MapStore.getMapModel(MapSetting.sMapId).flatTileLayer;
      this.satTileLayer = MapStore.getMapModel(MapSetting.sMapId).satTileLayer;
      this.focusLayer = MapStore.getMapModel(MapSetting.sMapId).focusLayer;
      this.markersLayer = MapStore.getMapModel(MapSetting.sMapId).markersLayer;
    }
    this.updateProps(this.props);
  }

  afterRenderMap() {        ...      // After the Leaflet map instance is initiated.

  // Helper function to switch different tile maps based on a flux MapStore value.
  renderMapTile() {
    if (MapStore.getMapTile(MapSetting.sMapId) == MAPTILE.FLAT) {
      if (!this.map.hasLayer(this.flatTileLayer)) {
        this.flatTileLayer.addTo(this.map);
        this.map.removeLayer(this.satTileLayer);
      }
    } else if (MapStore.getMapTile(MapSetting.sMapId) == MAPTILE.SATELLITE) {
      if (!this.map.hasLayer(this.satTileLayer)) {
        this.map.addLayer(this.satTileLayer);
        this.map.removeLayer(this.flatTileLayer);
      }
    }
  }

  // Helper function to create currently selected marker using svg layer.
  // @param location L.LatLng latitute and longitude of the marker location.
  renderFocusMarker(location) {
    if (location) {
      if (this.focusLayer.getLayers().length == 0) {
        let marker = createFocusMarker(location);
        this.focusLayer.addLayer(marker);
      } else {
        this.focusLayer.getLayers()[0].setLatLng(location);
      }
    }
  }

  // Executed everytime there are changes in the component.
  // @param nextProps Object updated properties.
  componentWillReceiveProps(nextProps) {
    this.updateProps(nextProps);
    if (nextProps.selected != this.props.selected) {
      this.renderPopup(TreeStore.getTree(nextProps.selected));
    }
  }

  // Helper function to apply new properties of the component.
  // @param props component properties.
  updateProps(props) {       ...
```

```
    // Helper function to render popup box for a selected marker.
    // @param tree Tree tree object.
    renderPopup(tree) {        ...

    // Helper function to render the currently selected marker.
    // @param tree Tree tree object.
    renderActiveMarker(tree) {       ...

    // Helper function to render markers on the leaflet map.
    // @param trees Array<Tree> collection of trees need to be rendered on the map.
    // @param selected Tree currently selected tree instance.
    renderMarkers(trees, selected) {
      if (__DEV__) {       // This value is coming from webpack's ENV value.
        console.log(`Map Tree - renderMarkers() with ${trees.length} trees`);
      }
      let markers = this.markersLayer.getLayers();
      // Remove unnecessary markers.
      for (let i = 0; i = markers.length;) {
        let bFound = false;     // Define value as false first and change once it finds item.
        if (markers[i].options.type == "svg" && markers[i].options.id == selected) {
          bFound = true;
        }
        if (markers[i].options.type == "canvas")  bFound = false;
        if (!bFound) {
          this.removeMarker(markers[i], this.markersLayer);
          markers = _.without(markers, markers[i]);     // Take out element from the array.
          i--;
        }
        i++;
      }
      // Add new markers.
      trees.forEach((tree) => { // ECMAScript 6 syntax for iterating each item in an array.
        let bFound = false;
        for (let i = 0; i = markers.length && !bFound; i++) {
          if (tree.id == markers[i].options.id) {
            bFound = true;
          }
        }
        if (tree.id != 0 && !bFound) {
          this.addMarker(tree, selected, false);
        }
      });
    }
    addMarker(tree, selected, editable) {   ...   // Helper function for adding markers.
    // Helper function for deleting markers.
    removeMarker(marker, layer) {
      marker.off('click'); // Call marker off event listener manually to close the popup.
      layer.removeLayer(marker);      // Remove the marker from the map layer.
    }

    // React render function.
    render () {
      return null;  // Override react rendering function with leaflet one.
    }
  }
```

calcsseason.php

```
/**
  @file: calcseason.php
  @purpose: Calculate season food based on history donation and rating of tree notes.
  @author Karl Kim (captainwhale52@gmail.com)
*/

<?php
  // Set headers not to cache the result of this php file.
  header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
  header("Cache-Control: post-check=0, pre-check=0", false);
  header("Pragma: no-cache");

  include_once 'functions.php';     // Import pdo helper functions.
  switch($_SERVER['REQUEST_METHOD']){     // Check the type of HTTP request.
    case 'GET':
      read(); break;
  }

  function read() {
    // Set time varialbed based on today's date.
    $startthisyear = new DateTime('-1 MONTH', new DateTimeZone('America/Los_Angeles'));
    $endthisyear = new DateTime('+1 MONTH', new DateTimeZone('America/Los_Angeles'));
    ...     // Define other time intervals of last year and 2 years ago.
    try {
      $pdo = getConnection(); // Initialze pdo connection with mysql database.
      $stmt = $pdo->prepare($sql);
      $stmt->execute();
      $sql = "SELECT id FROM `tree`";   // Construct the sql statement to fetch all trees.
      try {
        $pdo = getConnection();
        $stmt = $pdo->prepare($sql);
        $stmt->execute();
        $result1 = $stmt->fetchAll();
        foreach ($result1 as $tree) {
          // Construct the sql for finding tree items that has notes of the season.
          $sql = "SELECT `rate` FROM `note` WHERE `type` = 2
              AND (`tree` = " . $tree["id"] . ")
              AND ( (`date` BETWEEN '" . $startthisyear->format('Y-m-d') .
                  "' AND '" . $endthisyear->format('Y-m-d') . "')
              OR (`date` BETWEEN '" . $startlastyear->format('Y-m-d') .
                  "' AND '" . $endlastyear->format('Y-m-d') . "')
              OR (`date` BETWEEN '" . $start2lastyear->format('Y-m-d') .
                  "' AND '" . $end2lastyear->format('Y-m-d') . "') )
              ORDER BY `date` DESC LIMIT 1";
          try {
            $stmt = $pdo->prepare($sql);  // Pass sql statement into pdo.
            $stmt->execute();
            $result2 = $stmt->fetchAll();
            foreach ($result2 as $note) {
              $sql = "UPDATE `tree` SET `rate` = " . $note["rate"] .
                  " WHERE (`id` = " . $tree["id"] . ")";
              try {     // TODO: Combine all nested try and catch statement into one.
```

```
                $stmt = $pdo->prepare($sql);    $stmt->execute();
              } catch(PDOException $e) {  // Error handling.
                return '{"error":{"text":'. $e->getMessage() .'}}';
              }
            }
          } catch(PDOException $e) { return '{"error":{"text":'. $e->getMessage() .'}}'; }
        }
      } catch(PDOException $e) { return '{"error":{"text":'. $e->getMessage() .'}}'; }
    } catch(PDOException $e) { return '{"error":{"text":'. $e->getMessage() .'}}'; }
    // Finding a list of trees which have any donation data around the season.
    $sql = "SELECT tree.food FROM donate INNER JOIN tree on
        FIND_IN_SET(tree.id, donate.tree) INNER JOIN food on tree.food = food.id
        WHERE ABS(MOD(datediff(CURRENT_DATE, donate.date), 365)) >= 351
        GROUP BY tree.id ORDER BY ABS(MOD(datediff(CURRENT_DATE, donate.date), 365)) DESC";
    try {
      $pdo = getConnection(); $stmt = $pdo->prepare($sql);
      $stmt->execute(); $result = $stmt->fetchAll();
      $trees = [];      // Place holders for storing trees that meet the criteria.
      foreach ($result as $tree) {
        array_push($trees, $tree['food']);
      }
      if (sizeof($trees) > 0) {
        // Set the season flas as true if there is any tree meets the criteria.
        $sql = "UPDATE `food` SET `season` = 1 WHERE `id` IN (".implode(",", $trees).")";
        try {
          $pdo = getConnection();
          $stmt = $pdo->prepare($sql);
          $stmt->execute();
        } catch(PDOException $e) {
          return '{"error":{"text":'. $e->getMessage() .'}}';
        }
        $sql = "UPDATE `food` SET `season`=0 WHERE `id` NOT IN (".implode(",",$trees).")";
        try {
          $pdo = getConnection();   $stmt = $pdo->prepare($sql);  $stmt->execute();
          $params = array(    // Return code with 200 (success).
            "code" => 200,
          );
          echo json_encode($params);
        } catch(PDOException $e) {
          return '{"error":{"text":'. $e->getMessage() .'}}';
        }
      } else {
        // Set the season flag as false if there is no single meets the criteria.
        $sql = "UPDATE `food` SET `season` = 0";
        try {
          $pdo = getConnection();   $stmt = $pdo->prepare($sql);  $stmt->execute();
          $params = array(
            "code" => 200,
          );
          echo json_encode($params);
        } catch(PDOException $e) {
          return '{"error":{"text":'. $e->getMessage() .'}}';
        }
      }     ...   // Continue...
?>
```

leaflet-canvas-marker.js

```
/**
  @file: leaflet-canvas-marker.php
  @purpose: Custom marker class extends L.Circle to render a custom Canvas-based marker on
the leaflet map layer.
  @author Karl Kim (captainwhale52@gmail.com)
*/

(function (window, document, undefined) {  // JavaScript module define.
L.CanvasMarker = L.Circle.extend({
  statics: {
      //CLIP_PADDING: 0.02, // Not sure if there’s need to set it as a small value.
      CANVAS: true,
      SVG: false
  },
  // Initialize the marker class.
  initialize: function (latlng, radius, options) {
      L.Path.prototype.initialize.call(this, options);
      // Custom class variables for holding marker information.
      this._latlng = L.latLng(latlng);
      this._mRadius = radius;
      // Custom class variables for holding marker graphic assets.
      this._image = this.options.image;
      this._shadow = this.options.shadow;
      this._checkMode = this.options.checkMode;
      this._checked = this.options.checked;
  },

  _drawPath: function () {
    this._radius = this._map.getZoom() * 0.75;
    var p = this._point;
    if (this._image) {
      var width = this._map.getZoom();
      var height = Math.floor(width * 32 / 20);
      this._ctx.save(); // Save the matrix first before making any change.
      this._ctx.globalAlpha = this._map.getZoom() / 28;
      this._ctx.drawImage(this._shadow, p.x - height * 0.5
          + Math.floor(this._ctx.globalAlpha * 10), p.y - height, height, height);
      this._ctx.restore();     // Restore the saved matrix.
      // Add check mode graphic indicator based on the value.
      if (this._checkMode) {
        if (this._checked) {
          this._ctx.save();    // Save the matrix first before making any change.
          this._ctx.globalAlpha = 1;
          this._ctx.drawImage(this._image, p.x - width / 2, p.y - height, width, height);
          this._ctx.restore();       // Restore the saved matrix.
          this._ctx.drawImage(this._checked, p.x - width / 2, p.y - height, width,height);
        } else {  ...
      } else {    ...
    }
  },
});
}(window, document));
```

notify.php

```
/**
  @file: notify.php
  @purpose: get parameters from the client and send e-mails to parents who adopt trees
using Mailgun api and Ajax.
  @author Karl Kim (captainwhale52@gmail.com)
*/

<?php
  include_once 'functions.php';
  require_once("mailconfig.php");     // Import Maingun API.
  // Include the Autoloader (see "Libraries" for install instructions)
  require '../vendor/autoload.php';
  use Mailgun\Mailgun;
  switch($_SERVER['REQUEST_METHOD']){
    case 'POST':
      toparents();       break;
    case 'GET':
      read();      break;
  }
   // Find tree items which are on season, and return as json format to the client.
  function read() {      ...

  function toparents() {      // Send e-mails to parents who adopt season trees.
    $recipients = array();
    $config = new mailconfig();
    // Instantiate the mailgun object using configurations from the server.json file.
    $mg = new Mailgun($config->apikey);
    // Detect the base url of the server.
    $settings = json_decode(file_get_contents("../dist/setting/server.json"), true);
    $test = filter_var($settings['bTestMail'], FILTER_VALIDATE_BOOLEAN);
    $baseurl = $settings['ssltype'].$_SERVER['SERVER_NAME'].$settings['uBaseForRouter'];
    $treesurl = $baseurl."tree/";
    $gtolib = floatval($settings['fGToLBS']);
    // Compose sql statement to find trees and parents who adopt these trees.
    $sql = "SELECT tree.id, tree.parent, food.name, donate.date, donate.amount,
        donate.tree FROM donate INNER JOIN tree on FIND_IN_SET(tree.id, donate.tree)
        INNER JOIN food on tree.food = food.id WHERE `public` = 1
        AND ABS(MOD(datediff(CURRENT_DATE, donate.date), 365)) >= 335
        AND tree.id IN (" . $_POST['treeIds'] . ")
        GROUP BY tree.id ORDER BY ABS(MOD(datediff(CURRENT_DATE, donate.date), 365)) DESC";
    try {
      $pdo = getConnection(); $stmt = $pdo->prepare($sql);
      $stmt->execute(); $result = $stmt->fetchAll();
      $uparents = array();
      foreach ($result as $item) {
        // Break the string into array of parent ids.
        $parents = explode(",", $item['parent']);
        foreach ($parents as $parent) {
          if ($parent != 0)
            array_push($uparents, $parent);
        }
      }
```

```
      $uparents = array_unique($uparents);      // Remove redendant parent ids.

      // Send email to each parent.
      foreach ($uparents as $uparent) {
        $text = “”;
        $html = “”;
        foreach ($result as $item) {
          $parents = explode(“,”, $item[‘parent’]);
          if (in_array($uparent, $parents)) {
            $html .= “<div style=’font-family:sans-serif;margin-bottom:8px;’>
                   Please update “ . $item[‘name’] . “ #” . $item[‘id’]
                .= “ - <a href=’” . $treesurl . $item[‘id’] . “’>Post a new note for “
                . $item[‘name’] . “ #” . $item[‘id’] . “</a></div>
                   <img style=’max-width: 100%; height:auto;’ src=’” . $baseurl
                . “content/maps/” . $item[‘id’] . “_map.jpg’><br/><hr />”;
          }
        }

        // Retrieve mail address from a parent id.
        $sql = “SELECT contact FROM person WHERE `id` = “ . $uparent;
        try {
          $pdo = getConnection();   $stmt = $pdo->prepare($sql);
          $stmt->execute();   $result3 = $stmt->fetchAll();

          // Send the message.
          $mg->sendMessage($config->domain,
            array(‘from’     => $config->from,
                  ‘to’       => $result3[0][‘contact’],
                  ‘subject’  => “[TEST] FoodParent “ . date(“F j, Y”)
                                  . “ Update Request”,
                  ‘html’     => “<html><head></head><body><h3 style=’font-family:
                                    sans-serif;margin-bottom:16px;’>[TEST] FoodParent “
                                    . date(“F j, Y”) . “ Update Request</h3>” . $html
                                    . “</body></html>”,
            ));
          }
        } catch(PDOException $e) {
          $params = array(
            “code” => 400,     // Error code for the failure of sending e-mails.
          );
          echo json_encode($params);
        }
      }
      $params = array(
        “code” => 200,
      );
      echo json_encode($params);
    } catch(PDOException $e) {
      $params = array(
        “code” => 400,
      );
      echo json_encode($params);
    }
  }
?>
```

Project 2

Smart City Sensor Kit

Jong Won (Karl) Kim

project_overview.txt

The Public Design Workshop team at Georgia Institute of Technology have conducted smart city workshops at Atlanta in 2016, to gather public opinions about the smart city. For a part of hands-on activities of the workshop, I've built a standalone Arduino-based sensor box that participants can play with real-time data to understand the idea of smart cities.

project_members.txt

- Carl DiSalvo (Project Manager, Associate professor, School of Literature, Media, and Communication, Georgia Institute of Technology),
- Jong Won (Karl) Kim (Softeware Developer)
- Natalie Larkins (Prototype Designer)

my_contributions.txt

- Built an Arduino board and connect various type of sensors to measure environmental factors, including noise, temperature, humidity, altitude, Carbon Monoxide, dust level, and brightness.
- Connected a battery pack so that the box can work as a standalone platform for measuring environmental factors outside.
- Connected a real-time clock module on the board with a coin cell battery to keep track of time even though the box is turned off.
- Connected a SD / MMC card slot to store these measured values into an MicroSD card with timestamps.
- Connected a TFT screen to display measured values in real-time, and show various statuses of the box.

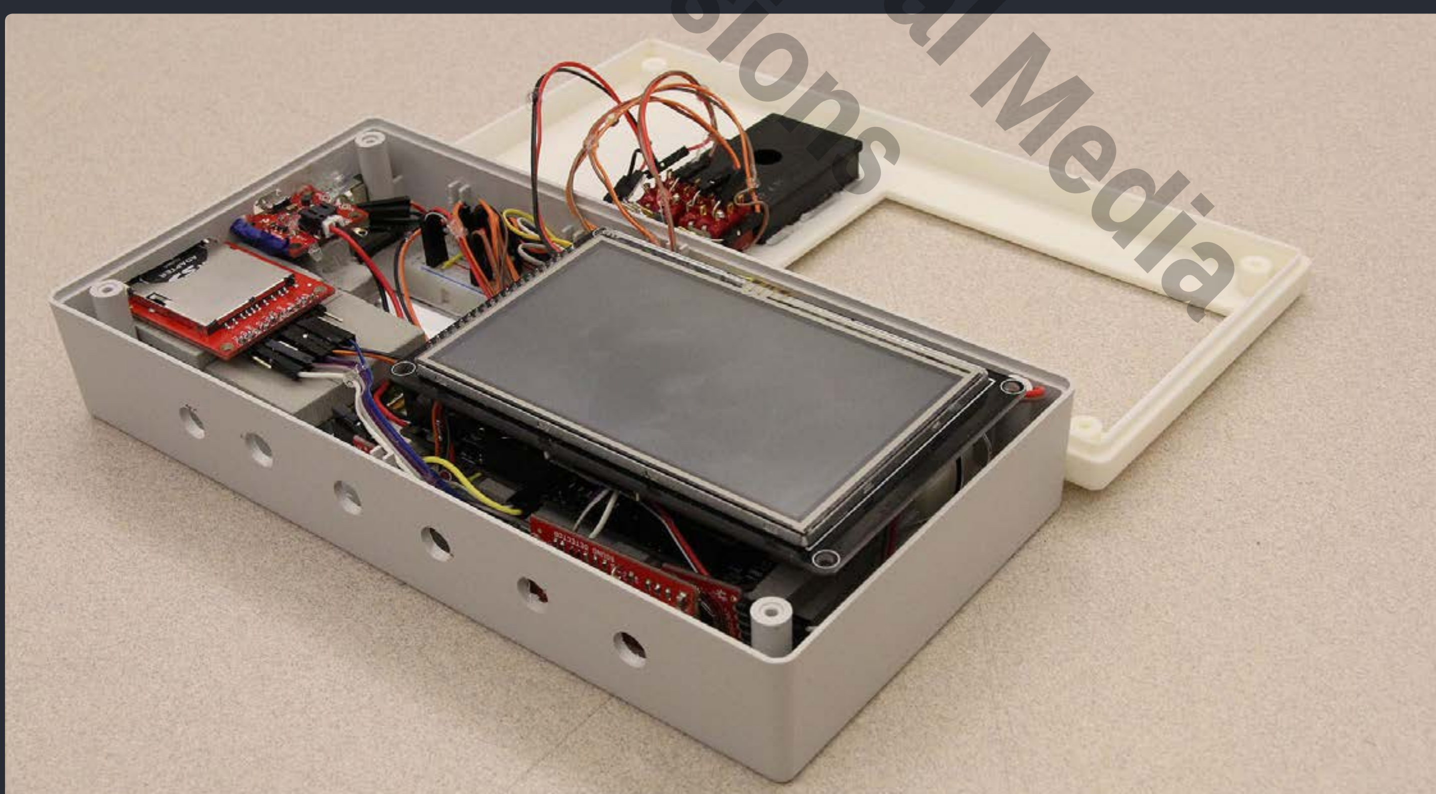
project_outcomes.html

- *Smart City Sensor Kit Guide - http://thecaptainwhale.com/thecaptainwhale/wp-content/uploads/2017/02/SmartCitySensorKitGuide.pdf*
- *PDW Smart City Platform Github Repository - https://github.com/PublicDesignWorkshop/PDWSmartCity*

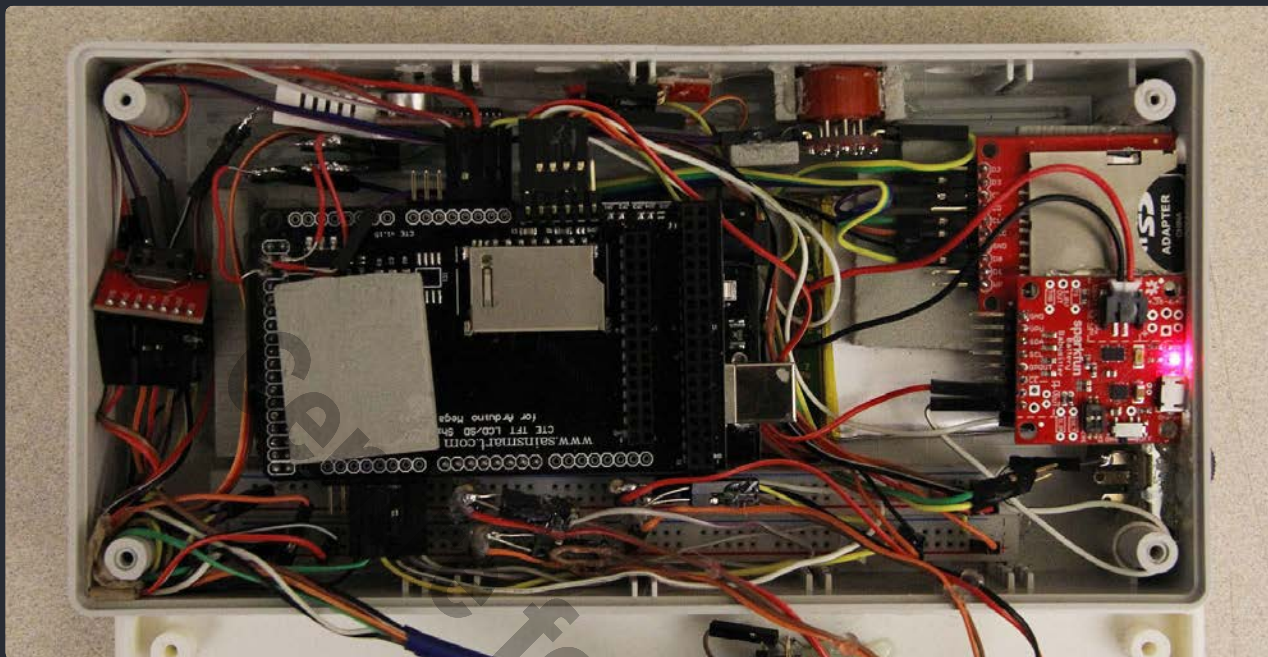
 Jong Won (Karl) Kim captainwhale52@gmail.com

sensor_box_top.jpg

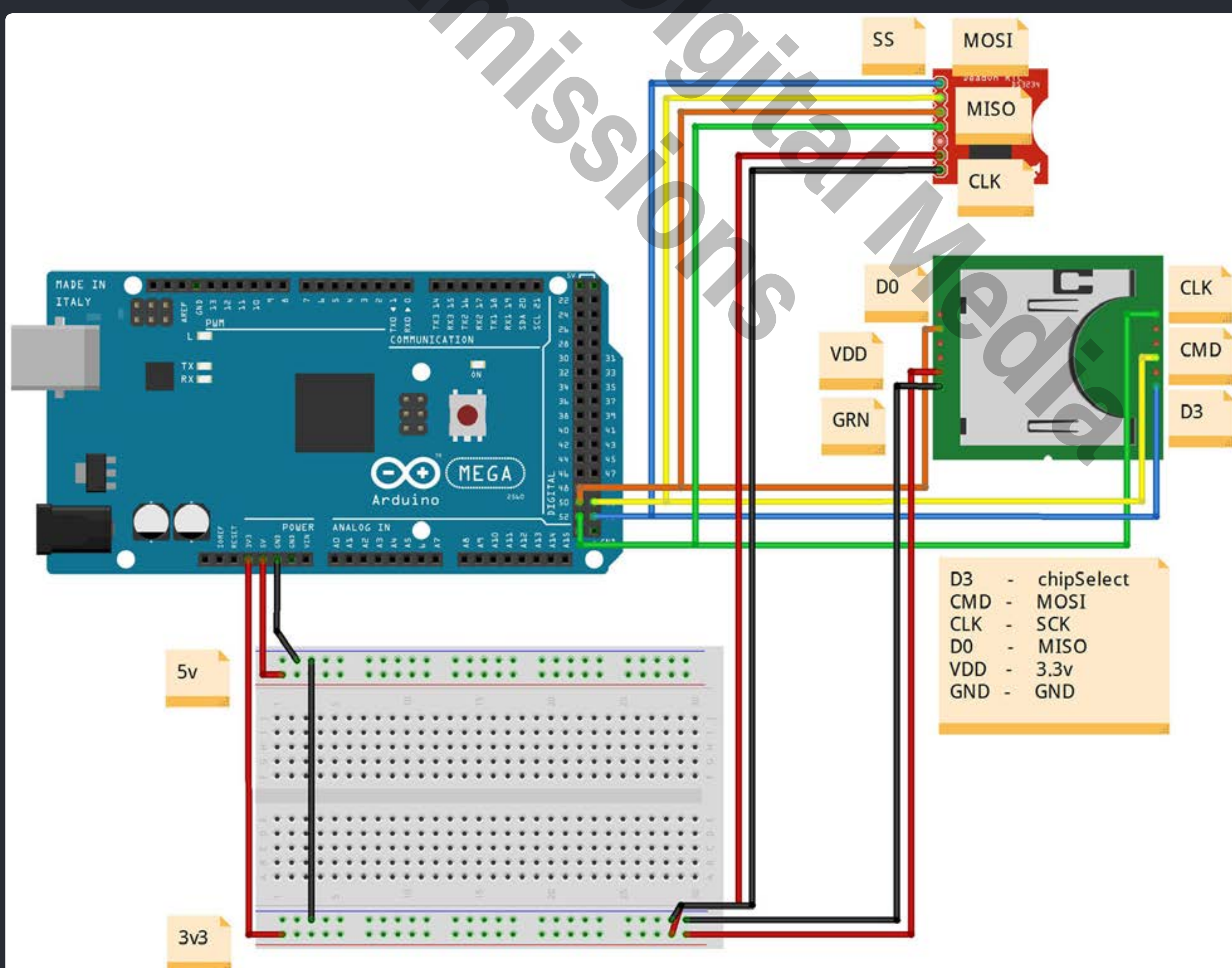
sensor_box_inside_1.jpg



sensor_box_inside_2.jpg



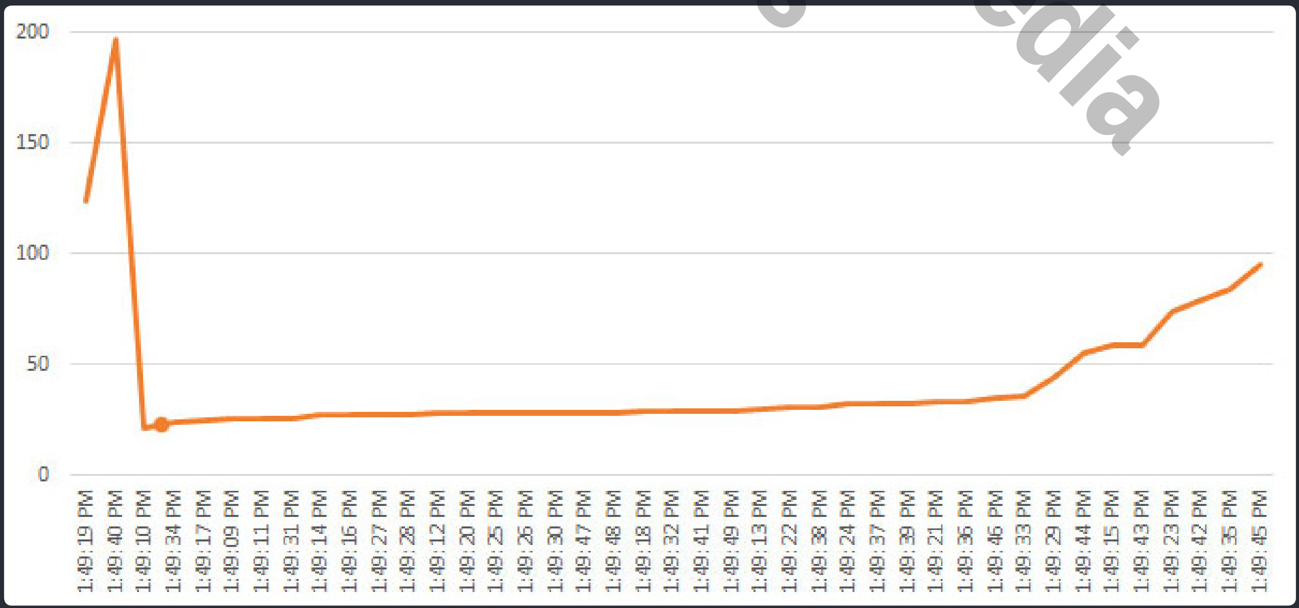
sensor_spi_wiring.jpg



sample_data_on_sdcard.jpg

TYPE	SENSOR MODEL	VALUE	UNIT	TIMESTAMP	LATITUDE	LONGITUDE
noise	SEN-12642	28	dB	1480686560	null	null
brightness	TEMT6000	47	lux	1480686560	null	null
temperature	RHT03	79	F	1480686560	null	null
humidity	RHT03	72	%	1480686560	null	null
dust	GP2Y1010AU0F	0	mg	1480686560	null	null
co	MQ-7	0	ppm	1480686560	null	null
noise	SEN-12642	33	dB	1480686561	null	null
brightness	TEMT6000	44	lux	1480686561	null	null
temperature	RHT03	79	F	1480686561	null	null
humidity	RHT03	72	%	1480686561	null	null
dust	GP2Y1010AU0F	0	mg	1480686561	null	null
co	MQ-7	0	ppm	1480686561	null	null
noise	SEN-12642	31	dB	1480686562	null	null
brightness	TEMT6000	37	lux	1480686562	null	null
temperature	RHT03	79	F	1480686562	null	null
humidity	RHT03	72	%	1480686562	null	null
dust	GP2Y1010AU0F	0	mg	1480686562	null	null
co	MQ-7	0	ppm	1480686562	null	null
noise	SEN-12642	74	dB	1480686563	null	null
brightness	TEMT6000	37	lux	1480686563	null	null
temperature	RHT03	79	F	1480686563	null	null
humidity	RHT03	72	%	1480686563	null	null
dust	GP2Y1010AU0F	0	mg	1480686563	null	null
co	MQ-7	0	ppm	1480686563	null	null
noise	SEN-12642	32	dB	1480686564	null	null
brightness	TEMT6000	38	lux	1480686564	null	null
temperature	RHT03	79	F	1480686564	null	null

noise_graph.jpg



sensor_with_tft.ino

```
/**
  @file: sensor_with_ttf.ino
  @purpose: Initialize sensor modules, sd card, real-time clock module, display sensor
values on the TFT display, and record measured values on the sd card.
  @author Karl Kim (captainwhale52@gmail.com)
*/

#include <Time.h>
#include <TimeLib.h>
...   // Import dependent libraries.

#define NOISE_IN A0
#define BRIGHTNESS_IN A1
#define RCT_CS_PIN 53
#define RCT_CS_PIN A2
#define RCT_CS_PIN A3
#define RCT_CS_PIN 18
#define RCT_CS_PIN A4
...   // Define Arduino communication pin numbers.

// Define struct for holding sensor values.
typedef struct {
  int sample; // the current sample number (0 ~ 9)
  time_t timestamp;
  int noise[GRAPH_SIZE];
  int temperature[GRAPH_SIZE];
  int humidity[GRAPH_SIZE];
  int brightness[GRAPH_SIZE];
  int dust[GRAPH_SIZE];
  int co[GRAPH_SIZE];
} MEASURES;

// Define struct for holding a list of sensors attached on the box.
typedef struct {
  int active; // The index number of the sensor which is currently selected.
  String types[SENSOR_SIZE];
  String units[SENSOR_SIZE];
  long mins[SENSOR_SIZE];
  long maxs[SENSOR_SIZE];
  String models[SENSOR_SIZE];
} SENSORS;

// Set up variables using the SD utility library functions.
Sd2Card card;
SdVolume volume;
SdFile root;
File dataFile;

void setup() {
   // Setting serial port.
   Serial.begin(9600);
   while(!Serial){} // Wait for until Arduino ready.
```

```
    pinMode(RTC_PIN, OUTPUT);  // RTC data pin number.
    digitalWrite(RTC_PIN,HIGH);      // Activate RTC.

    SPI.begin();
    // Sets Arduino in SPI master mode (for sharing SPI bewtween RTC and SD card).
    pinMode(SD_CARD, OUTPUT);

    // SPI communication is shared between RTC and SD card. Unlike I2C Bus communication,
  we need to manually switch mode to enable only one communication at one time.
    SPI.setDataMode(SPI_MODE0);      // Sets mode of SPI as 0 (for SD card communication).
    // Initialize SD card.
    if (card.init(SPI_HALF_SPEED, SD_CARD)) {
      if (!SD.begin(SD_CARD)) {
        is_sd_card_in = false;
      } else {
        is_sd_card_in = true;
      }

      // check the number of data-*.pdw files
      File root = SD.open("/");
      total_records = 0;
      while (true) {
        File entry = root.openNextFile();
        if (!entry) {
          // No more files.
          break;      // Escape from the while loop.
        } else if (((String) entry.name()).endsWith(".PDW")
            || ((String) entry.name()).endsWith(".pdw")) {
          total_records++;  // Increase the number not to override existing files.
        }
      }
    }

    // Now set the data mode of SPI as 3 for enabling RTC communication.
    SPI.setDataMode(SPI_MODE3);
    // Start Real-Time Clock (RTC)
    rtc.enable();
    rtc.begin(RTC_PIN);
    rtc.set12Hour();
    //  rtc.autoTime(); // This function import the time from your compiler machine and
  store into RTC module. You need to only use this command when you are first using the
  module, or after chaning a coin battery of the module.

    // Set the data mode of SPI as 0.
    SPI.setDataMode(SPI_MODE0);
    // Store all the latest 20 measured values of each sensor.
    measures = {   ...
    // Store information of sensors for showing on the LCD and storing data on the SD card.
    sensors = {    ...
    is_recording = false;
    total_snapshots = 0;

    // Set pin modes for interface buttons.
    pinMode(SELECT_PREV_SENSOR, INPUT);
```


Jong Won (Karl) Kim captainwhale52@gmail.com

```
    pinMode(SELECT_NEXT_SENSOR, INPUT);
    pinMode(TOGGLE_RECORD, INPUT);
    pinMode(DUST_LED_PIN, OUTPUT);

    // Call rht.begin() to initialize the sensor and our data pin
    rht.begin(RHT03_DATA_PIN);
    // Initialize TFT LCD screen
    myGLCD.InitLCD();
    myGLCD.clrScr();

   // Render currently selected sensor name
   renderActiveSensorName();
}

void loop() {
    static int8_t lastSecond = -1;
    // Call rtc.update() to update all rtc.seconds(), rtc.minutes(), etc. return functions.
    SPI.setDataMode(SPI_MODE3);
    rtc.update();
    if (rtc.second() != lastSecond) { // If the time has passed.
       renderTime(); // Render a new time.
       lastSecond = rtc.second(); // Update lastSecond value.

       // Measure value for each sensor (measure functions should be executed every loop();
however, for now, I simply measure each sensor value every second, because there is some
measure delay issue).
       measureNoise();
       measureBrightness();
       measureDust();
       measureCO();
       measureTemperatureAndHumidity();

       // Unrender prevous graph.
       unrenderActiveSensorGraph();

       // Store new values into the measures array.
       updateNoise();
       updateBrightness();
       updateTemperature();
       updateHumidity();
       updateDust();
       updateCO();

       // Render new values on the display.
       renderActiveSensorGraph();
       renderActiveSensorValue();

       // If file is open (recording is active), write sensor values on the file is open.
       if (dataFile) {
          for (int i = 0; i < SENSOR_SIZE; i++) {
             String result = (String) sensors.types[i] + " ";
             result.toLowerCase();
             result += sensors.models[i] + " ";
             if ((String) sensors.types[i] == "NOISE" ) {
                result += (String) measures.noise[GRAPH_SIZE-2] + " ";
```

```
            } else if ((String) sensors.types[i] == “BRIGHTNESS”) {
               result += (String) measures.brightness[GRAPH_SIZE-2] + “ “;
            } else if ((String) sensors.types[i] == “TEMPERATURE”) {
               result += (String) measures.temperature[GRAPH_SIZE-2] + “ “;
            } else if ((String) sensors.types[i] == “HUMIDITY”) {
               result += (String) measures.humidity[GRAPH_SIZE-2] + “ “;
            } else if ((String) sensors.types[i] == “DUST”) {
               result += (String) measures.dust[GRAPH_SIZE-2] + “ “;
            } else if ((String) sensors.types[i] == “CO”) {
               result += (String) measures.co[GRAPH_SIZE-2] + “ “;
            } else {
               result += “null “;
            }
            result += sensors.units[i] + “ “;
            result += (String) measures.timestamp + “ “;
            result += “ null null”;
            // Write result string on the last part of the file.
            dataFile.println(result);
         }
      }
   }
   // SPI mode back to 0.
   SPI.setDataMode(SPI_MODE0);
   // Functions to catch the pressed status of UI buttons.
   detectActiveSensor();
   detectRecordStatus();
   detectNewSnapshot();

   delay(100);    // Delay 100 milliseconds before executing the next loop function.
}

// Measure an optical dust sensor value and convert into ppm.
void measureDust() {
   digitalWrite(DUST_LED_PIN, LOW);
   delayMicroseconds(DUST_SAMPLING_TIME);
   float voMeasured = analogRead(DUST_MEASURE_PIN);
   delayMicroseconds(DUST_DELTA_TIME);
   digitalWrite(DUST_LED_PIN, HIGH);
   delayMicroseconds(DUST_SLEEP_TIME);
   float calcVoltage = voMeasured * (3.3 / 1024);
   // linear eqaution taken from http://www.howmuchsnow.com/arduino/airquality/
   // Chris Nafis (c) 2012
   float dustDensity = (0.17 * calcVoltage - 0.1) * 1000;
   if ( dustDensity < 0) {
      dustDensity = 0.00;
   }
   measures.dust[GRAPH_SIZE-1] = dustDensity;
}

void updateDust() {
   for (int i = 0; i < GRAPH_SIZE-1; i++) {
      measures.dust[i] = measures.dust[i+1];
   }
   measures.dust[GRAPH_SIZE-1] = 0;
}     ...   // Continue...
```


Jong Won (Karl) Kim captainwhale52@gmail.com

Project 3

Drones for Foraging

ANDROID JAVA

Jong Won (Karl) Kim

project_overview.txt

Drones for Foraging is a design project that received notable mentions in the Core77 Design Award student speculative category. The project includes an Android application prototype that communicates with a drone to send it to a designated locations, and take photos from trees to monitor fruit ripening of the trees using a simple OpenCV object detection algorithm.

project_members.txt

- Carl DiSalvo (Project Manager, Associate professor, School of Literature, Media, and Communication, Georgia Institute of Technology),
- Caroline Foster (Design Researcher)
- Jong Won (Karl) Kim (Softeware Developer)
- Tasmia Alam (Design Researcher)
- Tom Jenkins (Design Researcher)

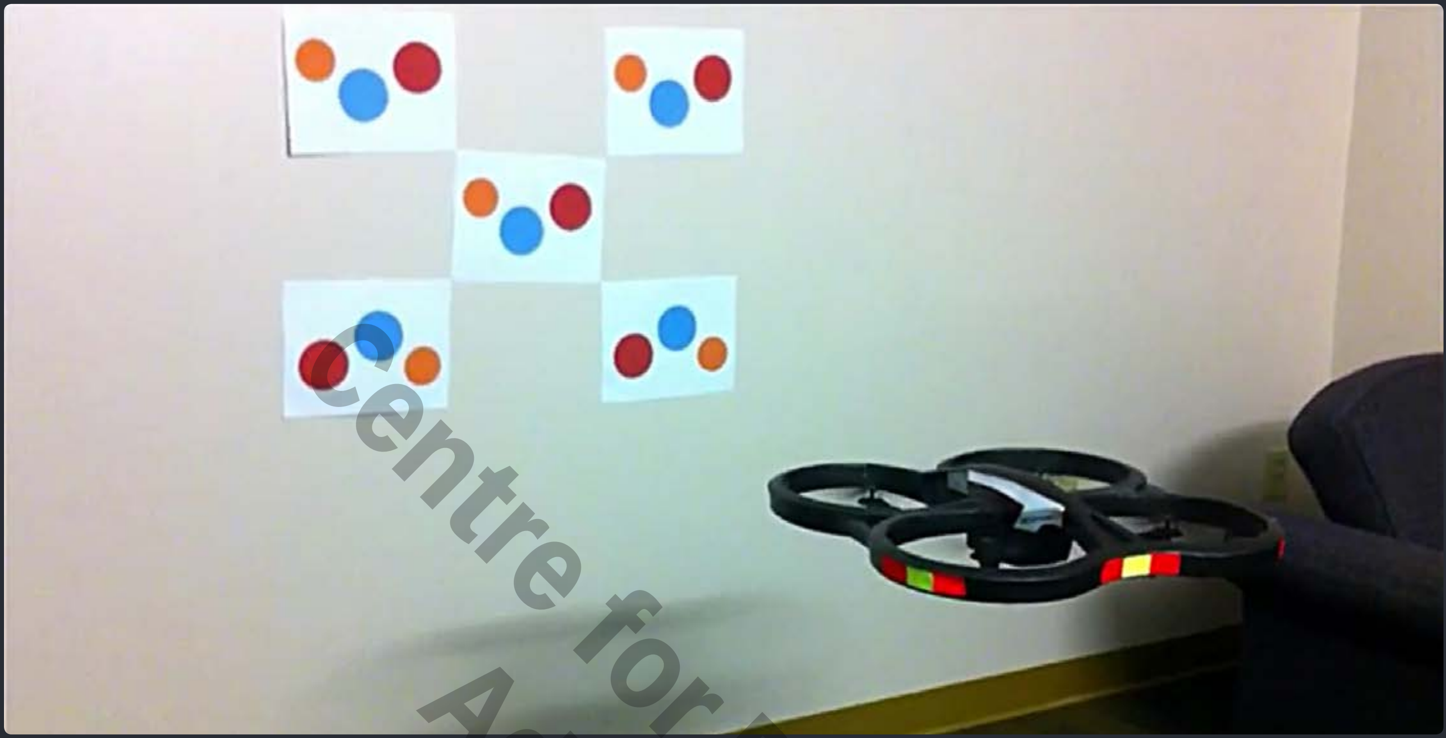
my_contributions.txt

- Analyzed the *Parrot AR. Drone 2.0* quadrocopter API document to understand the AT Commands and the system flow of the Parrot.
- Imported the Shigeo YOSHIDA's *ARDroneForP5* library into the project to create the socket network interface system.
- Decoded the video stream from the Parrot and render on Android.
- Applied the Hough Circle Object Detection Algorithm as an initial attemp of detecting red apples.
- Built an auto navitagation mode using Nutiteq's Geographic Information System (GIS) map library and a GPS module.

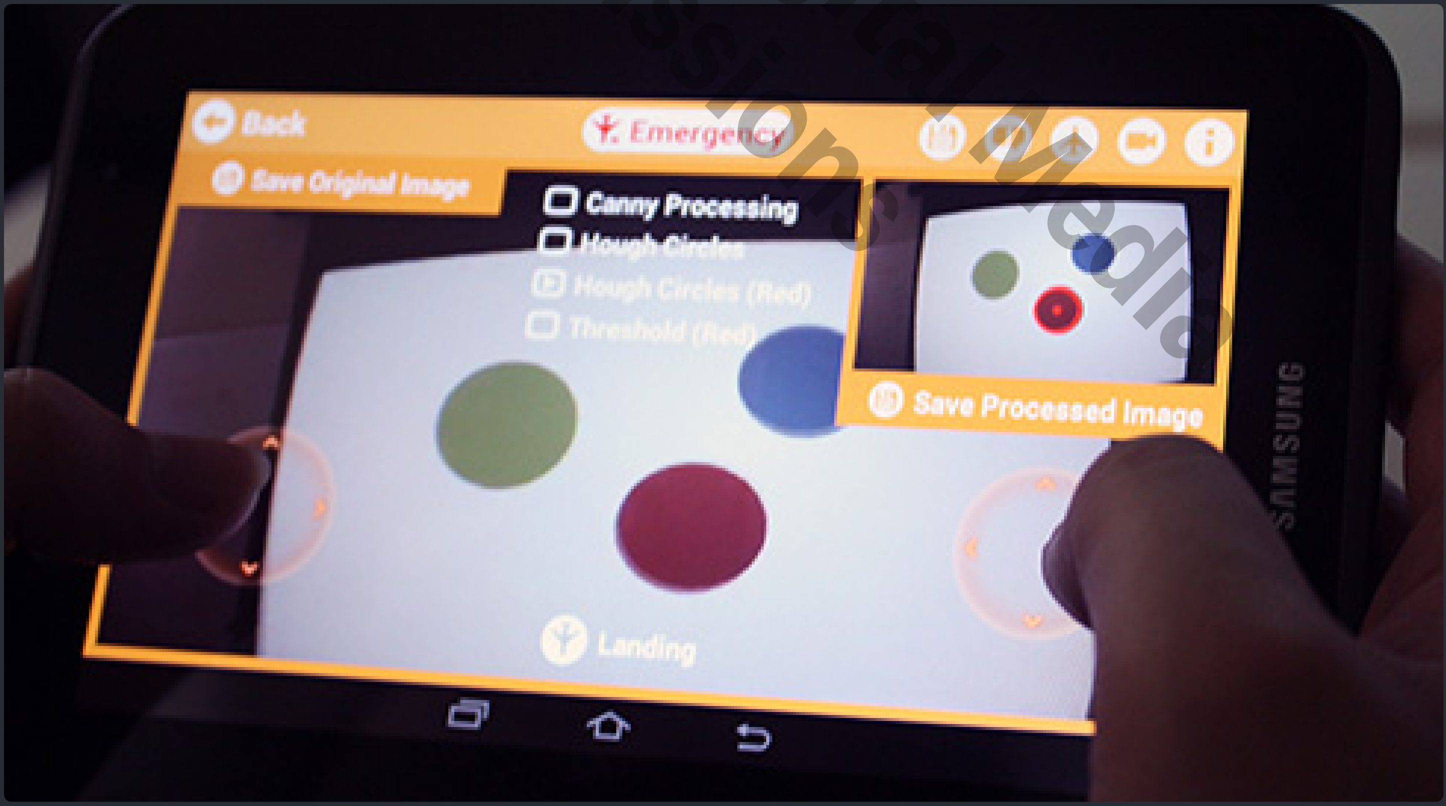
project_outcomes.html

- *Drones for Foraging Core77 Award - http://www.core77designawards.com/2014/recipients/drones-foraging*
- *Android Application Prototype Github Repository - https://github.com/happykarl/ForagingTech*
- *Android Application Prototype Auto-navigation Demo - https://youtu.be/NWHXcDzO0xw*
- *Android Application Prototype Red Circle Detection Demo - https://youtu.be/UTSpA_ww8Rs*

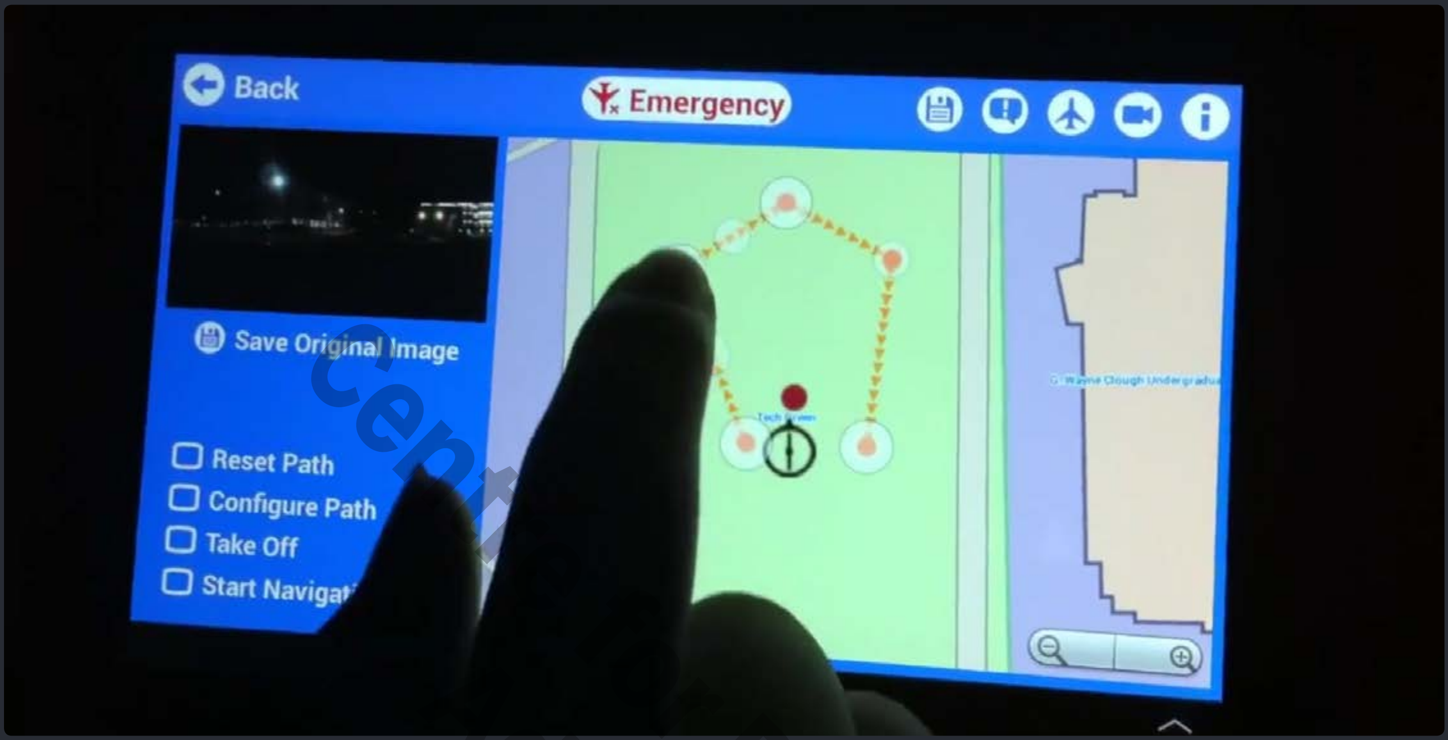
red_circle_detection.jpg



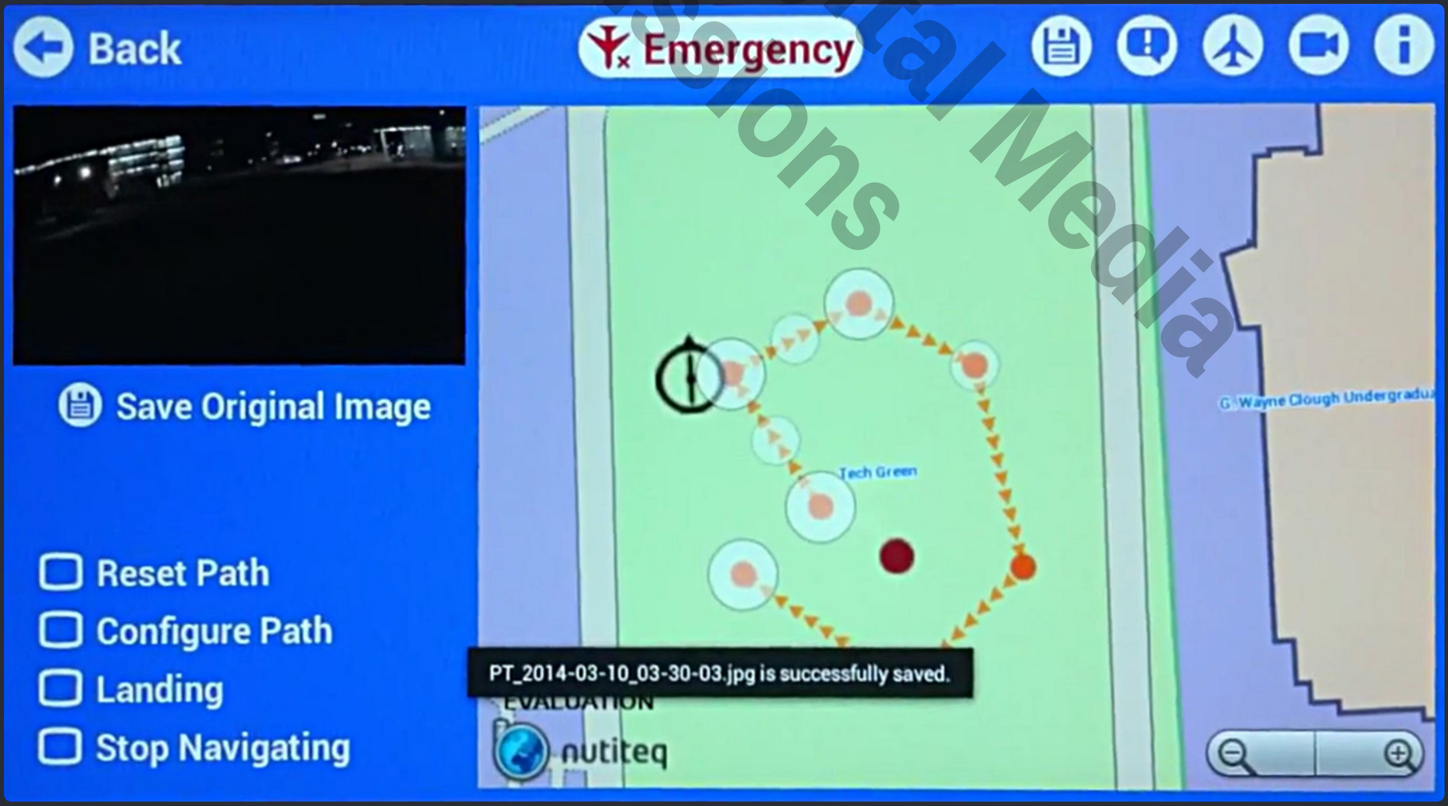
manual_flying.jpg



navigating_setup_path.jpg



navigating_save_image.jpg



CameraFragment.java

```java
/**
   @file: CameraFragment.java
   @purpose: Render a video feed from a drone and apply object detection algorithms.
   @author Karl Kim (captainwhale52@gmail.com)
*/

package com.felicekarl.foragingtech.views.fragments;
// Import opencv libraries.
import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

// Import the Android built-in surface texture listener to render video frames into it.
import android.view.TextureView.SurfaceTextureListener;

public class CameraFragment extends BaseFragment
      implements TextureView.SurfaceTextureListener,
      OnClickListener, UpdateCameraFragmentButtonListener {
   ... // Define class variables.
   private RenderThread mThread;    // Use a separate thread for rendering video frames.
   private Bitmap video;
   private int imageWidth;
   private int imageHeight;
   private long startMs;      // For calculating the framerate of the video.
   // OpenCV variables.
   private Point pt;    // point storing a screen location of detected circle.
   private Object mFrameSyncObject = new Object();
   private boolean mFrameAvailable = true;
   public enum IMAGEMODE{      // To store the current image processing mode.
      CANNY, HOUGHCIRCLES, REDHOUGHCIRCLES, REDTHRESHOLD
   }
   private IMAGEMODE imgMode;
   @Override
   // Create surface with the video stream
   // @param surface SurfaceTexture video texture
   // @param width int width of the video
   // @param height int height of the video
   public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {
     mSurfaceTexture = surface;
     mSurface = new Surface(mSurfaceTexture);
     startMs = System.currentTimeMillis();
   }
   @Override
   // Called back every time the a video frame is updated.
   // @param surface SurfaceTexture new video frame
   public void onSurfaceTextureUpdated(SurfaceTexture surface) {
      if (camMode != null && camMode.equals(CAMERAMODE.FLYING)) {
```

```
                if ( (System.currentTimeMillis() - startMs) > 50 ) {
                    // mTextureView is holding a raw video frame.
                    video = mTextureView.getBitmap(imageWidth, imageHeight);
                    startMs = System.currentTimeMillis();
                }
            }
        }
        // Custom SurfaceTextureListner to plug the image processing middleware.
        // @param surface SurfaceTexture new video frame
        private class CanvasListener implements SurfaceTextureListener {
            @Override
            // Override the frame call back function to plugin a rendering thread.
            public void onSurfaceTextureAvailable(SurfaceTexture surface,int width,int height) {
                imageWidth = width;
                imageHeight = height;
                mThread = new RenderThread();    // Create video rendering thread after.
                mThread.start();
            }
            @Override
            // Destroy the rendering thread.
            public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
                if (mThread != null) {
                    mThread.stopRendering();    // Destory video rendering thread before.
                }
                return true;
            }
        }
        // Custom Thread class for handling the OpenCV image processing.
        private class RenderThread extends Thread {
            // Define status variable as volatile to be shared across threads.
            private volatile boolean mRunning = true;
            private long prevMS = 0;

            @Override
            // Run function will be executed whenever the thread is active.
            public void run() {
                // Safe while loop by checking the validity of the thread.
                while (mRunning && !Thread.interrupted( )) {
                    if ( (System.currentTimeMillis() - prevMS) > 50
                            && video != null && !video.isRecycled() ) {
                        final Canvas canvas = mImageView.lockCanvas(null);
                        try {
                            if (imgMode.equals(IMAGEMODE.CANNY)) {    ...
                            } else if (imgMode.equals(IMAGEMODE.HOUGHCIRCLES)) {  ...
                            } else if (imgMode.equals(IMAGEMODE.REDHOUGHCIRCLES)) {
                                Mat sourceMat = new Mat(imageWidth, imageHeight, CvType.CV_8UC1);
                                Mat thresholdMat = new Mat(imageWidth, imageHeight, CvType.CV_8UC1);
                                Utils.bitmapToMat(video, sourceMat);
                                Utils.bitmapToMat(video, sourceMat);
                                // Change color mode into HSV.
                                Imgproc.cvtColor(sourceMat, hsvMat, Imgproc.COLOR_RGB2HSV, 4);
                                // Filter out the image with redish color.
                                Core.inRange(hsvMat, new Scalar(-2, 80, 80),
                                        new Scalar(3, 255, 255), thresholdMat);
                                // Apply the guassian blur before applying the hough circle.
```

```
                        Imgproc.GaussianBlur(thresholdMat, thresholdMat, new Size(5,5),2,2);
                        Mat circles = new Mat();    // holding detected circles.

                        // TODO: expose these variables via Android UIs.
                        int iCannyUpperThreshold = 40;
                        int iMinRadius = 5;
                        int iMaxRadius = 400;
                        int iAccumulator = 100;
                        int iLineThickness = 2;
                        // Apply the hough circle and save the result into circles variable.
                        Imgproc.HoughCircles(thresholdMat, circles,
                                Imgproc.CV_HOUGH_GRADIENT, 2.0, thresholdMat.rows() / 4,
                                iCannyUpperThreshold, iAccumulator, iMinRadius, iMaxRadius);
                        if (circles.cols() > 0){
                            for(int i = 0; i < circles.cols(); i++){
                                double vCircle[] = circles.get(0,i);
                                if (vCircle == null) break;
                                // If there is any detected circle.
                                pt = new Point(Math.round(vCircle[0]), Math.round(vCircle[1]));
                                int radius = (int) Math.round(vCircle[2]);
                                // Draw the detected circle on the surface.
                                Core.circle(thresholdMat, pt, radius,
                                    new Scalar(255,0,0), iLineThickness);
                                Core.circle(thresholdMat, pt, 2,
                                    new Scalar(255,0,0), iLineThickness);
                            }
                        }
                        // Swap the final video frame with the original.
                        Utils.matToBitmap(thresholdMat, video);
                        // Release memory.
                        sourceMat.release();
                        thresholdMat.release();
                        circles.release();
                    }
                    canvas.drawBitmap(video, 0, 0, null);
                } finally {
                    if (video != null)       video.recycle();
                    mImageView.unlockCanvasAndPost(canvas);
                }
            }
            prevMS = System.currentTimeMillis();
        } else {
            try {
                Thread.sleep(10);     // Update video frame every 10 miliseconds.
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public void stopRendering() {
        interrupt();
        mRunning = false;
    }
}
...       // Continue...
```

GpsListener.java

```java
/**
   @file: GpsListener.java
   @purpose: Expose interface to accept different implementations for different usages. For
example, one implementation can be for displaying GPS information on the screen, or the
other implementation can be for operating the auto navigation of a drone.
   @author Karl Kim (captainwhale52@gmail.com)
*/

package com.felicekarl.ardrone.managers.navdata.listeners;

public interface GpsListener {
      void gpsLocChanged(double lat, double lon, double elevation, double hdop,
                  int data_available, double lat0, double lon0, double lat_fuse,
                  double lon_fuse, long gps_state, double vdop, double pdop,
                  float speed, long last_frame_timestamp, float degree,
                  float degree_mag);
}
```

NavDataParser.java

```java
/**
   @file: NavDataParser.java
   @purpose: Read the navigation data, such as magneto, velocity, and gps, and execute an
actual implementation of the GPSListener interface.
   @author Cliff L. Biffe., Karl Kim (captainwhale52@gmail.com)
*/

package com.felicekarl.ardrone.managers.navdata;

public class NavDataParser implements UpdateAttitudeListener, UpdateBatteryListener,
      UpdateGpsListener, UpdateMagnetoListener, UpdateStateListener,
      UpdateVelocityListener {
   ...      // Define listner holders.
   private GpsListener mGpsListener;
   /// The navigation data are as a buffer format to increase speed of data transaction.
   private void processNavDataGPS(ByteBuffer optionData) {
      double lat = optionData.getDouble();
      double lon = optionData.getDouble();
      double elevation = optionData.getDouble();
      double hdop = optionData.getDouble();
      int data_available = optionData.getInt();
      byte[] unk_0 = new byte[8];
         for (int i = 0; i < unk_0.length; i++) {
            unk_0[i] = optionData.get();
         }
         double lat0 = optionData.getDouble();
         double lon0 = optionData.getDouble();
         double lat_fuse = optionData.getDouble();
         double lon_fuse = optionData.getDouble();
         long gps_state = optionData.getInt() & 0xFFFFFFFF;
         byte[] unk_1 = new byte[40];
```

```
        for (int i = 0; i < unk_1.length; i++) {
            unk_1[i] = optionData.get();
        }
        double vdop = optionData.getDouble();
        double pdop = optionData.getDouble();
        float speed = optionData.getFloat();
        long last_frame_timestamp = optionData.getInt() & 0xFFFFFFFF;
        float degree = optionData.getFloat();
        float degree_mag = optionData.getFloat();

        if (mGpsListener != null && lat != 0.0d && lon != 0.0d) {
            // Call the implemted function of GpsLisnter
            mGpsListener.gpsLocChanged(lat, lon, elevation, hdop, data_available, lat0,
                    lon0, lat_fuse, lon_fuse, gps_state, vdop, pdop, speed,
                    last_frame_timestamp, degree, degree_mag);
        }
    }
    ...   // Continue...
```

MainPresenter.java

```
/**
   @file: MainPresenter.java
   @purpose: Main presenter initialize essential instances and implementations of
interfaces. The instance of ARDrone class is created, the acutal implementaiton of the
interface GPSListner is created, and these two things are connected together in this
presenter.
   @author Karl Kim (captainwhale52@gmail.com)
*/

package com.felicekarl.foragingtech.presenters;

...   // Import dependent classes.
import com.felicekarl.ardrone.managers.navdata.listeners.GpsListener;

public class MainPresenter implements Runnable {
   public boolean initDrone() {
      // Add a GPS listener.
      mARDrone.updateGpsListener(new GpsListener() {
         @Override
         public void gpsLocChanged(double lat, double lon, double elevation,
               double hdop, int data_available, double lat0, double lon0,
               double lat_fuse, double lon_fuse, long gps_state, double vdop,
               double pdop, float speed, long last_frame_timestamp, float degree,
               float degree_mag) {
            // Actual implementation of the gpsLocChanged.
            // Change the marker's location of the drone on the Nutiteq's map.
            view.setDroneCurPos(lat, lon);
            view.updateUserCurPos();
         }
      });
   }
}   ...   // Continue...
```

Project 4

Escape Goat 2 (GBA)

Jong Won (Karl) Kim

project_overview.txt

Escape Goat 2 (GBA) is a 2D puzzle game for Game Boy Advance (GBA) that I made for the final project of the Media Device Architecture class at Georgia Tech. The idea of the game is inspired from the original puzzle platform game, *Escape Goat 2*, on Steam. The game is complied as .gba format, which can be played using emulators on Mac / Windows, or actual devices (NDS, 3DS) using a special cartridge, called DSTwo.

my_contributions.txt

- Analyzed *TONC - GBA Programming* document to understand the hardware and software interface of GBA.
- Used double video buffers and tilemaps for rendering the 320x240 GBA screen resolution with 24 frames per second (fps).
- Used one sprite for animating the goat (character) actions, explosion effects, and platforms movements.
- Created 16-bit bitmap tilemaps and sprites from the original art works of *Escape Goat 2* (the permission is granted by the original developer).
- Used two sound queues (background and sfx) at the same time using a timer and hardware interrupts.
- Rendered two simultaneous backgrounds with different speeds to achieve the parallax scrolling (3D movement feeling) effect.
- Exposed a map editor interface for creating custom game levels.
- Used Raycast for detecting collisions between the goat and other environments (walls, moving platforms, and obstacles) around it.
- Compiled the game into .gba format by injecting a gba compiler.

project_outcomes.html

- *Escape Goat 2 (GBA) Rom (with Mac / Windows Emulators) - http://thecaptainwhale.com/thecaptainwhale/wp-content/uploads/2017/01/EscapeGoat2-GBA.zip*
- *Escape Goat 2 (GBA) Playthrough - https://youtu.be/dr47XjliBlw*

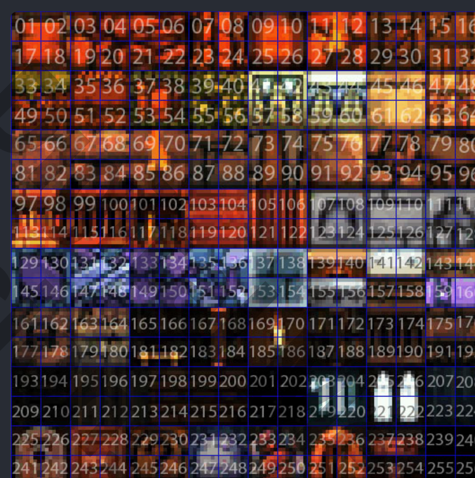
splash.jpg

instruction.jpg

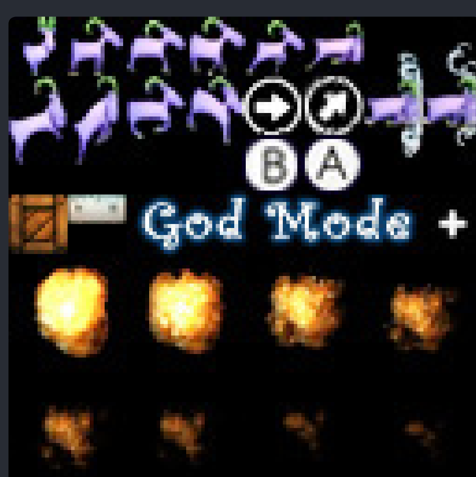
stage_1.jpg

nds_dstwo.jpg

tilemap.bmp



sprite.bmp



gameAsset.h

```c
/**
  @file: gameAsset.h
  @purpose: Define structures for the player character, platforms, and interactive objects,
as well as collision detection functions between the player and the game environment.
  @author Karl Kim (captainwhale52@gmail.com)
*/

typedef struct {  // Player structure to hold various statuses of the player.
  int hoff;  int voff;    int col;    float row;    int cd;    float rd;
  int width;      int height;    int sprite;
  int god_mode;   // Cheat mode.
  PLAYER_STATUS status;
  PLAYER_RAM_STATUS ram_prev_status;
  PLAYER_RAM_STATUS ram_cur_status;
  PLAYER_WALK_STATUS walk_prev_status;
  PLAYER_WALK_STATUS walk_cur_status;
  PLAYER_JUMP_STATUS jump_prev_status;
  PLAYER_JUMP_STATUS jump_status;
  PLAYER_JUMP2_STATUS jump2_prev_status;
  PLAYER_JUMP2_STATUS jump2_status;
  PLAYER_BOTTOM_HIT_STATUS bottom_hit_status;
  PLAYER_UPPER_HIT_STATUS upper_hit_status;
  PLAYER_SIDE_HIT_STATUS side_hit_status;
  PLAYER_GROUND_STATUS ground_status;
  SCROLL_STATUS scroll_status;
  DIRECTION direction;
  int ground_shaking;
} PLAYER;

typedef struct {  // Stucture of platform for checking collision detection.
  int col;
  int row;
  int orig_row;
  int direction;
  int width;
  int height;
  int layer;
  int animation;
} PLATFORM;

extern PLATFORM platforms[600];
extern PLATFORM objects[40];
extern int num_objects;

void detectGroundCollision();
PLATFORM *detectBottomCollision();
PLATFORM *detectSideCollision();
PLATFORM *detectUpperCollision();

...   // Continue...
```

gameAsset.c

```c
/**
  @file: gameAsset.c
  @purpose: Implementation of the functions defined in gameAsset.h file.
  @author Karl Kim (captainwhale52@gmail.com)
*/

int num_platform; // Number of platforms.
int num_objects;  // Number of collidable object except platforms.
PLATFORM platforms[];
PLATFORM objects[];
PLATFORM *bottom_hit_platform;      // Hoding the pointer address of bottom platform.
PLAYER player;
POINT p1, p2, p3, p4, p5, p6, p7, p8, p9;

void detectGroundCollision() {      ...   // Check whether the goat is on ground or not.

// Find the platform that the goat is hitting on its feet.
PLATFORM *detectBottomCollision() {
  player.bottom_hit_status = PLAYER_BOTTOM_HIT_OFF;
  int i;
  // Platform collision detection.
  for (i = 0; i < num_platform; i++) {
    if (platforms[i].layer != 0) {
      if ( ( (player.col + player.width/2 + SCREENWIDTH/2) > platforms[i].col)
          && ( (player.col + player.width/2 - SCREENWIDTH/2) < platforms[i].col ) ) {
        // p3 (top raycast) check
        if ( (p3.col >= (platforms[i].col)) && (p3.col <= (platforms[i].col +
            platforms[i].width - 1)) && (p3.row >= (platforms[i].row))
            && (p3.row <= (platforms[i].row + 1)) ) {
          player.bottom_hit_status = PLAYER_BOTTOM_HIT_ON;
              return &platforms[i];      // Return the pointer of the platform.
        } else if ( (p4.col >= (platforms[i].col + 1))
            && (p4.col <= (platforms[i].col + platforms[i].width))
            && (p4.row >= (platforms[i].row)) && (p4.row <= (platforms[i].row + 1)) ) {
          player.bottom_hit_status = PLAYER_BOTTOM_HIT_ON;
          return &platforms[i];     // return the pointer of the platform.
        } else if ( (p5.col >= (platforms[i].col + 1))
            && (p5.col <= (platforms[i].col + platforms[i].width))
            && (p5.row >= (platforms[i].row)) && (p5.row <= (platforms[i].row + 1)) ) {
          player.bottom_hit_status = PLAYER_BOTTOM_HIT_ON;
          return &platforms[i];
        }
      }
    }
  }
  ... // Object (non-platform) Collision Detection.
  return;   // Return null if nothing is on top of the goat.
}

PLATFORM *detectSideCollision() {   ...   // Check and return a side platform that is
touching with the goat.
```

```
PLATFORM *detectUpperCollision() {  ...   // Check and return a top platform that is
touching with the goat.
...   // Continue...

// Update the player based on the user's input or collision with platforms and objects.
void updatePlayer() {
  // Define positions of rays of the player. To save system resources, it only shoots one
pixels around the goat.
  p1.col = player.col;
  p1.row = player.row;
  p2.col = player.col + player.width;
  p2.row = player.row;
  p3.col = player.col;
  p3.row = player.row + player.height;
  p4.col = player.col + player.width;
  p4.row = player.row + player.height;
  p5.col = player.col + player.width / 2;
  p5.row = player.row + player.height;
  p6.col = player.col;
  p6.row = player.row + player.height / 2;
  p7.col = player.col + player.width;
  p7.row = player.row + player.height / 2;
  p8.col = player.col;
  p8.row = player.row + player.height - 2;
  p9.col = player.col + player.width;
  p9.row = player.row + player.height - 2;

  if (player.ram_cur_status == PLAYER_RAM_OFF) {
    if (player.status == PLAYER_STAY) {
      player.cd = 0;
    }
  }
  // Not to make jump in the middle air.
  player.jump_status = PLAYER_JUMP_ON;
  // Delete this part to make it possible to jump one time in the middle of air.

  PLATFORM *upper_hit_platform = detectUpperCollision();    // Return the pointer of
collided platform.
  if (player.upper_hit_status == PLAYER_UPPER_HIT_ON) {
    player.rd = 0;
    player.row = upper_hit_platform->row + upper_hit_platform->height + 1;
  }

  // Bottom collision detection.
  bottom_hit_platform = detectBottomCollision();
  if (player.bottom_hit_status == PLAYER_BOTTOM_HIT_ON) {
    player.rd = 0;
    player.row = bottom_hit_platform->row - player.height;
    player.jump_status = PLAYER_JUMP_OFF;
    player.jump_prev_status = PLAYER_JUMP_OFF;
    player.jump2_status = PLAYER_JUMP2_DEFAULT;
    player.jump2_prev_status = PLAYER_JUMP2_DEFAULT;
  } else {
    // If there is no platfom on the bottom of the goat, check whether the goat is in air.
    detectGroundCollision();
```

```
    if (player.ground_status == PLAYER_GROUND_OFF) {
      player.rd += 0.24;
      if (player.rd >= 1.4) {
        player.rd = 1.4;
      }
      player.row += player.rd;
    } else {
      player.jump_status = PLAYER_JUMP_OFF;
    }

    // Side collision detection.
    PLATFORM *side_hit_platform = detectSideCollision();
    if (player.side_hit_status == PLAYER_LEFT_HIT_ON) {
        ...
      player.ram_cur_status = PLAYER_RAM_OFF;      // Stop ramming if the player hit a wall.
    } else if (player.side_hit_status == PLAYER_RIGHT_HIT_ON) {      // Right collision.
      player.cd = 0;
      player.col = side_hit_platform->col - player.width;
      if (player.jump_status == PLAYER_JUMP_ON
          && player.jump2_status == PLAYER_JUMP2_DEFAULT) {
        player.jump2_status = PLAYER_JUMP2_OFF;
        player.jump2_prev_status = PLAYER_JUMP2_OFF;
      }
      if (side_hit_platform->layer == 2 && player.ram_cur_status == PLAYER_RAM_ON) {
        side_hit_platform->layer = -1;
        player.status = PLAYER_STUNNED;     // Stun the player once it hits a wall.
        // Add SFX sound.
        playSoundB(explosionSFX,EXPLOSIONSFXLEN,EXPLOSIONSFXFREQ, 0);
      }
      if (side_hit_platform->layer == 1 && player.ram_cur_status == PLAYER_RAM_ON) {
        player.status = PLAYER_STUNNED;
        player.ground_shaking = 1;    // Shake the entire screen once the ramming is done.
        // Add SFX sound.
        playSoundB(shakeSFX,SHAKESFXLEN,SHAKESFXFREQ, 0);
      }
      if (side_hit_platform->layer == 3 && player.ram_cur_status == PLAYER_RAM_ON) {
        player.status = PLAYER_STUNNED;
        player.ground_shaking = 1;    // Shake the entire screen once the ramming is done.
        // Add SFX sound.
        playSoundB(shakeSFX,SHAKESFXLEN,SHAKESFXFREQ, 0);
      }
      player.ram_cur_status = PLAYER_RAM_OFF;
    } else if (player.side_hit_status == PLAYER_SIDE_HIT_OFF) {
      // Bring the player back to the game if the goat goes outside of the game map.
      player.col += player.cd;
      if (player.col < 16) {
        player.col = 16;
      }
      if (player.row > GROUND_HEIGHT - 16) {
        player.row = GROUND_HEIGHT - 16;
      }
    }
  }
  ...   // Continue...
```

stage.c

```c
/**
  @file: stage.c
  @purpose: Store the map data with each number matches with the tile map index number.
Each number matches the index numbers on tilemap.bmp.
  @author Karl Kim (captainwhale52@gmail.com)
*/

// Left tile set of the stage (1024x240)
const unsigned short midgroundMap[] = {
   000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
    28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29,
   106,107, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99,
   122,123,114,115, 20, 21, 20, 21,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,
   104,105,102,103,  4,  5,  4,  5,  4,  5, 42, 43,  4,  5,  6,  7, 44, 45, 44, 45, 44, 45, 44, 45, 44, 45, 44, 45, 34, 35, 34, 35,
   120,121,118,119, 32, 33, 20, 21, 20, 21, 58, 59, 20, 21, 22, 23, 60, 61, 60, 61, 12, 13, 60, 61, 60, 61, 60, 61, 50, 51, 96, 97,
   104,105,  2,  3, 48, 49,  4,  5, 32, 33, 80, 81, 32, 33, 68, 69, 44, 45, 12, 13,000,000, 12, 13, 44, 45, 44, 45, 34, 35, 14, 15,
   120,121, 18, 19, 64, 65, 20, 21, 48, 49, 20, 21, 48, 49, 84, 85, 12, 13,000,000,000,000,000,000, 12, 13, 60, 61, 50, 51, 30, 31,
   104,105,  2,  3,  4,  5,  4,  5, 64, 65,  4,  5, 64, 65, 66, 67,000,000,000,000,000,000,000,000,000,000, 12, 13, 34, 35, 96, 97,
   120,121, 18, 19, 20, 21, 20, 21, 20, 21, 20, 21, 20, 21, 82, 83, 28, 29,000,000,000,000,000,000,000,000,000,000, 50, 51, 40, 41,
   104,105,  2,  3, 32, 33,  4,  5, 32, 33,  4,  5, 32, 33, 66, 67, 44, 45, 28, 29,000,000,000,000,000,000,000,000,110,111, 56, 57,
   120,121, 18, 19, 48, 49, 20, 21, 48, 49, 20, 21, 48, 49, 82, 83, 60, 61, 60, 61, 28, 29,000,000,000,000,000,000,126,127, 40, 41,
   104,105,  2,  3, 64, 65,  4,  5,  4,  5,  4,  5,  4,  5, 66, 67, 44, 45, 44, 45, 44, 45, 28, 29,000,000, 76, 77,110,111, 56, 57,
   120,121, 18, 19, 20, 21, 20, 21, 20, 21, 20, 21, 20, 21, 82, 83, 60, 61, 60, 61, 60, 61, 60, 61, 28, 29, 92, 93,126,127, 40, 41,
   104,105,  2,  3,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5, 66, 67, 44, 45, 44, 45, 44, 45, 44, 45, 44, 45, 76, 77, 34, 35, 56, 57,
   120,121, 18, 19, 20, 21, 20, 21, 20, 21, 20, 21, 20, 21, 82, 83, 60, 61, 60, 61, 60, 61, 60, 61, 60, 61, 92, 93, 50, 51, 40, 41,
   106,107,  2,  3, 36, 37,  4,  5, 68, 69,  4,  5,  4,  5, 66, 67, 76, 77, 76, 77, 44, 45, 44, 45, 44, 45, 76, 77, 34, 35, 56, 57,
   122,123, 18, 19, 52, 53, 20, 21, 84, 85, 20, 21, 20, 21, 82, 83, 92, 93, 92, 93, 60, 61, 60, 61, 60, 61, 92, 93, 50, 51, 40, 41,
    70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71,
    86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87,

// Right tile set of the stage (1024x240)
   000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,
    28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29, 28, 29,
    98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99, 98, 99,
   114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,114,115,
    50, 51, 50, 51, 50, 51,100,101, 44, 45, 44, 45, 44, 45,000,000,  2,  3,  4,  5,  4,  5,  4,  5,  4,  5,  4,  5,102,103,104,105,
    96, 97, 96, 97, 34, 35,116,117, 60, 61, 60, 61, 60, 61,000,000, 18, 19, 20, 21, 20, 21, 20, 21, 20, 21, 20, 21,118,119,120,121,
    14, 15, 14, 15, 50, 51, 44, 45, 44, 45, 44, 45, 12, 13, 68, 69,  2,  3, 32, 33,  4,  5,102,103,  4,  5, 32, 33,  6,  7,104,105,
    30, 31, 30, 31, 34, 35,100,101, 60, 61, 12, 13,000,000, 84, 85, 18, 19, 48, 49, 20, 21,118,119, 20, 21, 48, 49, 22, 23,120,121,
    96, 97, 96, 97, 50, 51,116,117, 12, 13,000,000,000,000, 66, 67,  2,  3, 80, 81,  4,  5,108,109,  4,  5,  4,  5,  6,  7,104,105,
    40, 41, 40, 41, 34, 35,100,101,000,000,000,000,000,000, 82, 83, 18, 19, 20, 21, 20, 21,124,125, 20, 21, 20, 21, 22, 23,120,121,
    56, 57, 56, 57, 50, 51,116,117,000,000,000,000,000,000, 66, 67,  2,  3, 32, 33,  4,  5,108,109,  4,  5, 32, 33,  6,  7,104,105,
   100,101, 40, 41, 34, 35,100,101,000,000,000,000,000,000, 82, 83, 18, 19, 48, 49, 20, 21,124,125, 20, 21, 48, 49, 22, 23,120,121,
   116,117, 56, 57, 50, 51,116,117,000,000,000,000,000,000, 66, 67,  2,  3, 80, 81, 20, 21,102,103,  4,  5,  4,  5,  6,  7,104,105,
   100,101,100,101,100,101,100,101,000,000,000,000,000,000, 82, 83, 18, 19, 20, 21, 20, 21,118,119, 20, 21, 20, 21, 22, 23,120,121,
   116,117,116,117,116,117,116,117,000,000,000,000, 28, 29, 66, 67,  2,  3,  4,  5,  4,  5,108,109,  4,  5,  4,  5,  6,  7,104,105,
    40, 41, 40, 41, 34, 35, 44, 45,000,000, 28, 29, 60, 61, 82, 83, 18, 19, 20, 21, 20, 21,124,125, 20, 21, 20, 21, 22, 23,120,121,
    56, 57, 56, 57, 50, 51, 60, 61, 28, 29, 44, 45, 44, 45, 66, 67, 64, 65,  4,  5,  4,  5,108,109,  4,  5, 36, 37,  6,  7,  8,  9,
    40, 41, 40, 41, 34, 35,100,101, 60, 61, 60, 61, 60, 61, 82, 83, 18, 19, 20, 21, 20, 21,124,125, 20, 21, 52, 53, 22, 23, 24, 25,
    70, 71, 70, 71, 70, 71,116,117, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71, 70, 71,
    86, 87, 86, 87, 86, 87, 70, 71, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87, 86, 87
}      ...   // Continue...
```

Project 5

The Kiln - Masonry Texture Generator

WEBPACK JS NODE.JS PYTHON BLENDER

Jong Won (Karl) Kim

WEBPACK JS NODE.JS PYTHON BLENDER

project_overview.txt

Masonry is one of the oldest industry in the world which historically builds structures from individual units bound by mortar. These units are used on buildings for aesthetic reasons as opposed to structural. *The Kiln*, the parametric brick texture generator, provides enhanced experience to architects for generating photo-realistic brick textures using a dedicated Python server and 3D computer graphic software, called Blender. Through mobile devices, architects can connect to the site and create different stories of masonry units in real-time.

project_clients.txt

- Dr. Russell Gentry (Associate Professor, School of Architecture, Georgia Institute of Technology)

project_members.txt

- Krystina Madej (Project Advisor, Associate professor, Georgia Institute of Technology)
- June Park (Project Manager)
- Brenda Lin (UX/UI Designer)
- Jong Won (Karl) Kim (Lead Software Developer, UX/UI Designer)
- Tre'Saun Thomas (Director, Writer)

my_contributions.txt

- Built a python server using Flask framework for the network communication between clients and Blender.
- Built a responsive web application using React framework.
- Created a server-side rendering pipeline using Ajax pattern.
- Created a cycle-based Blender material and expose texture parameters using a Python script.
- Uploaded and ran the project via Amazon Web Server (AWS).

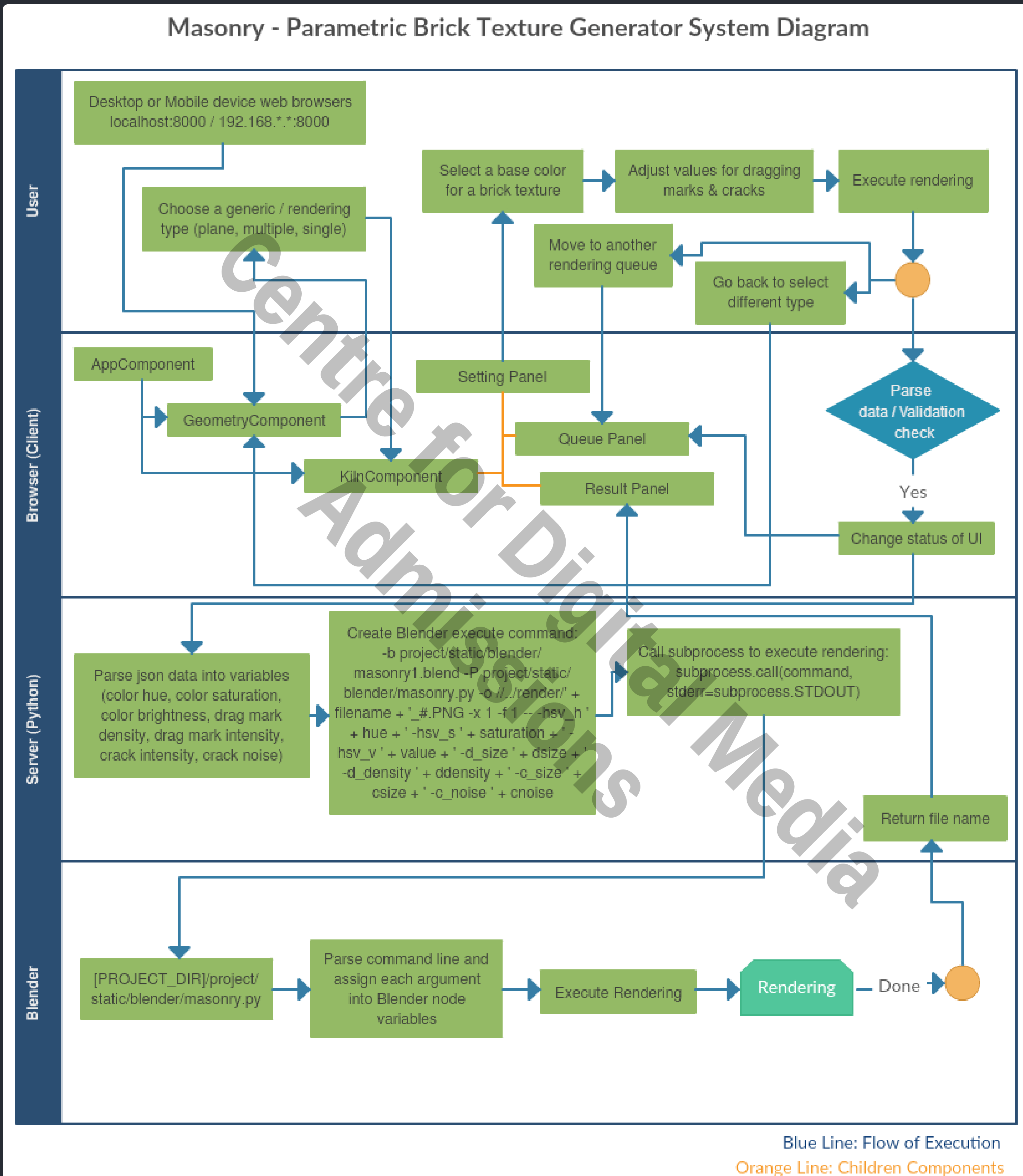
project_outcomes.html

- *The Kiln App - https://thecaptainwhale.com/thekiln* (It will take about 1 minute to render textures due to the free server speed.)
- *The Kiln Demo Video - https://youtu.be/p6AGkW37kO8*
- *The Kiln Github Repository - https://github.com/captainwhale52/Masonry*

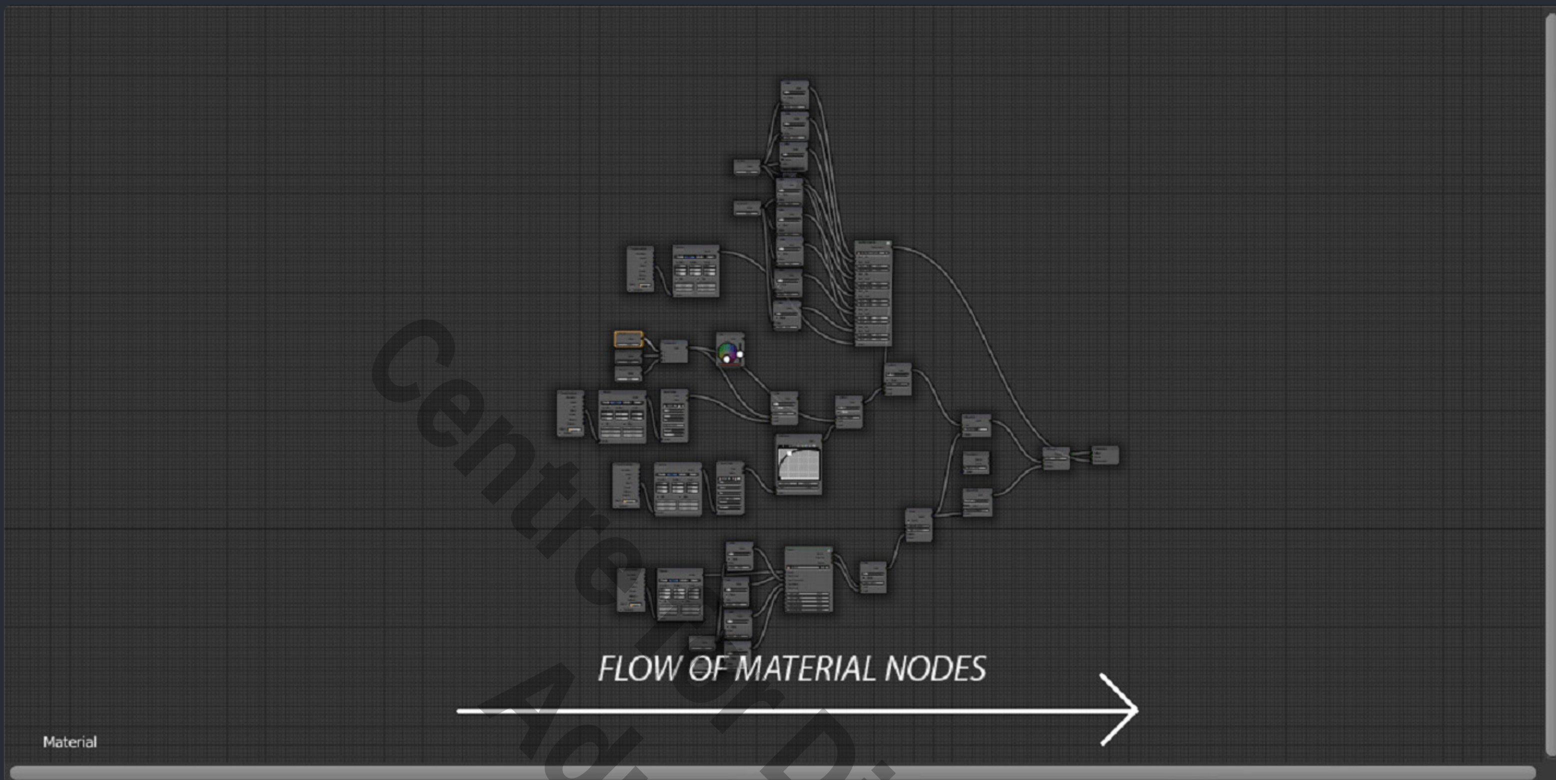
Jong Won (Karl) Kim captainwhale52@gmail.com

WEBPACK JS NODE.JS PYTHON BLENDER

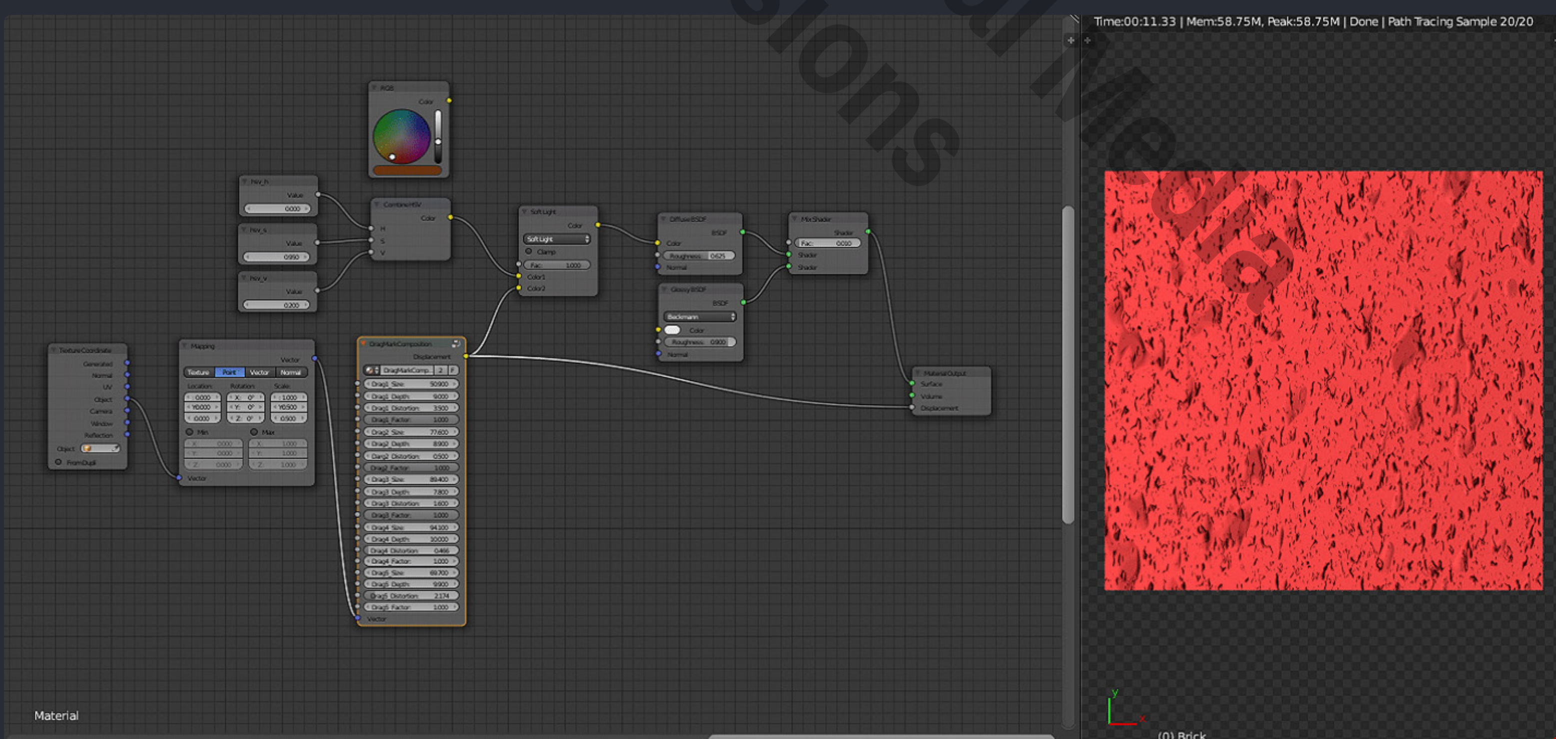
system_diagram.jpg



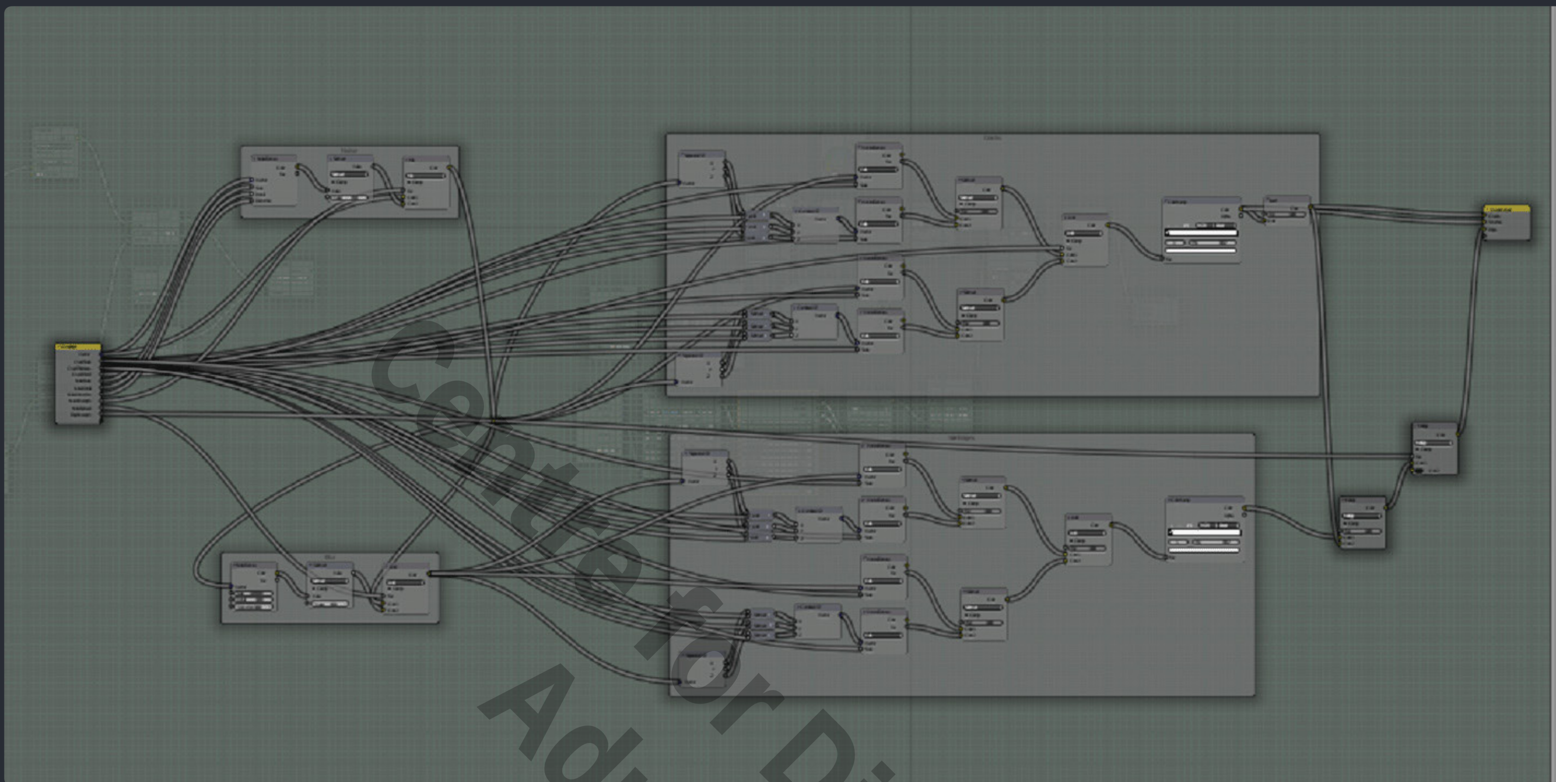
material_overview.jpg



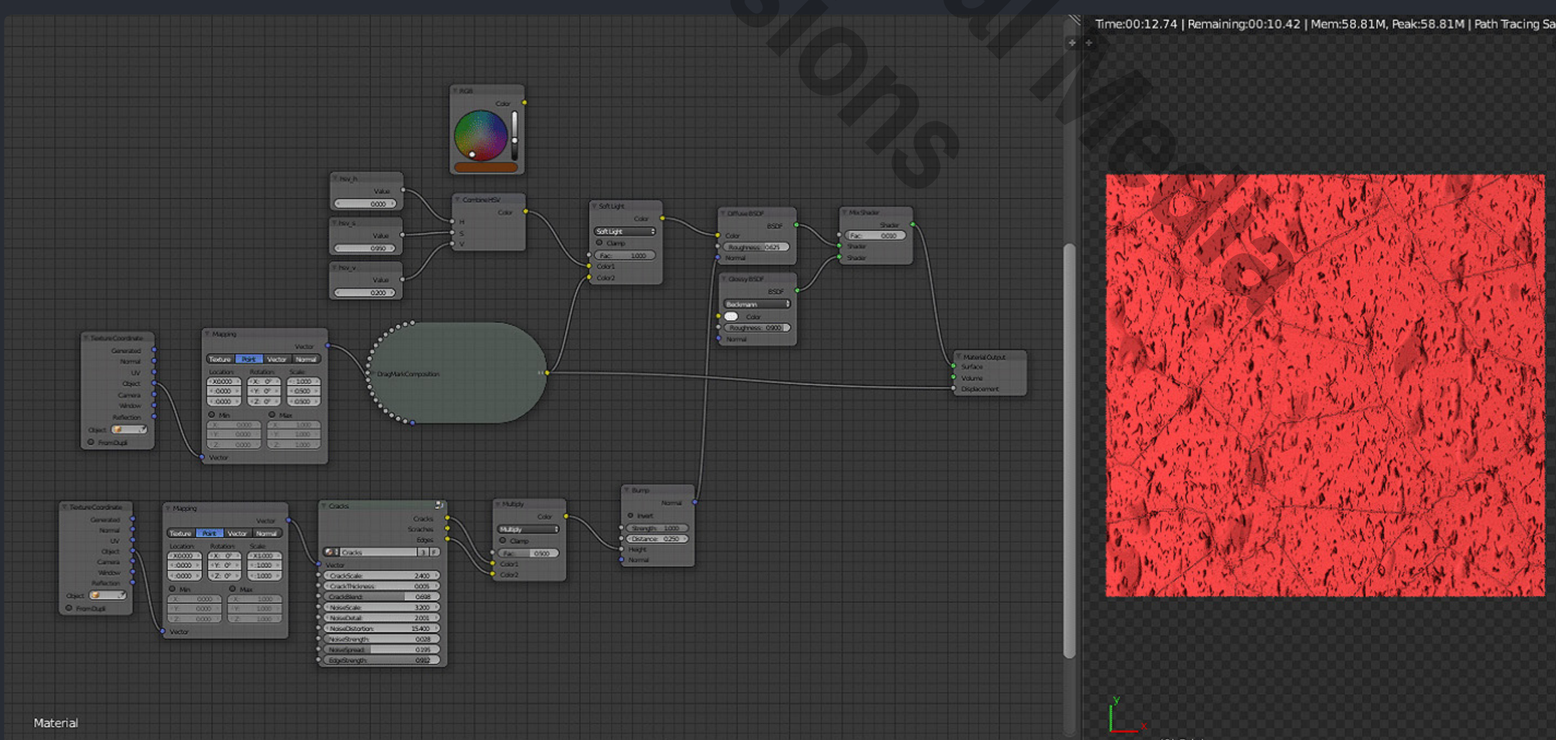
material_dragmark.jpg



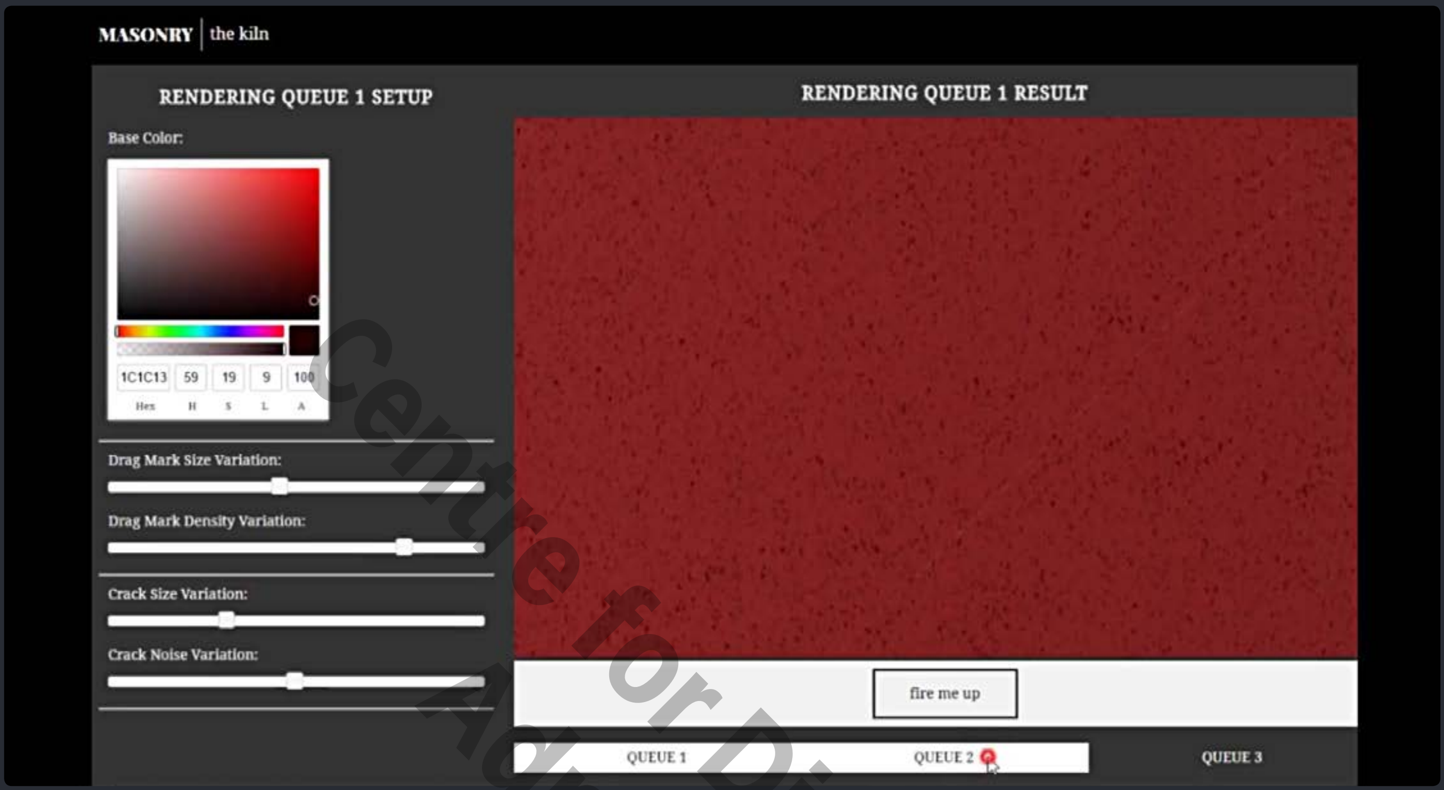
material_crack_node.jpg



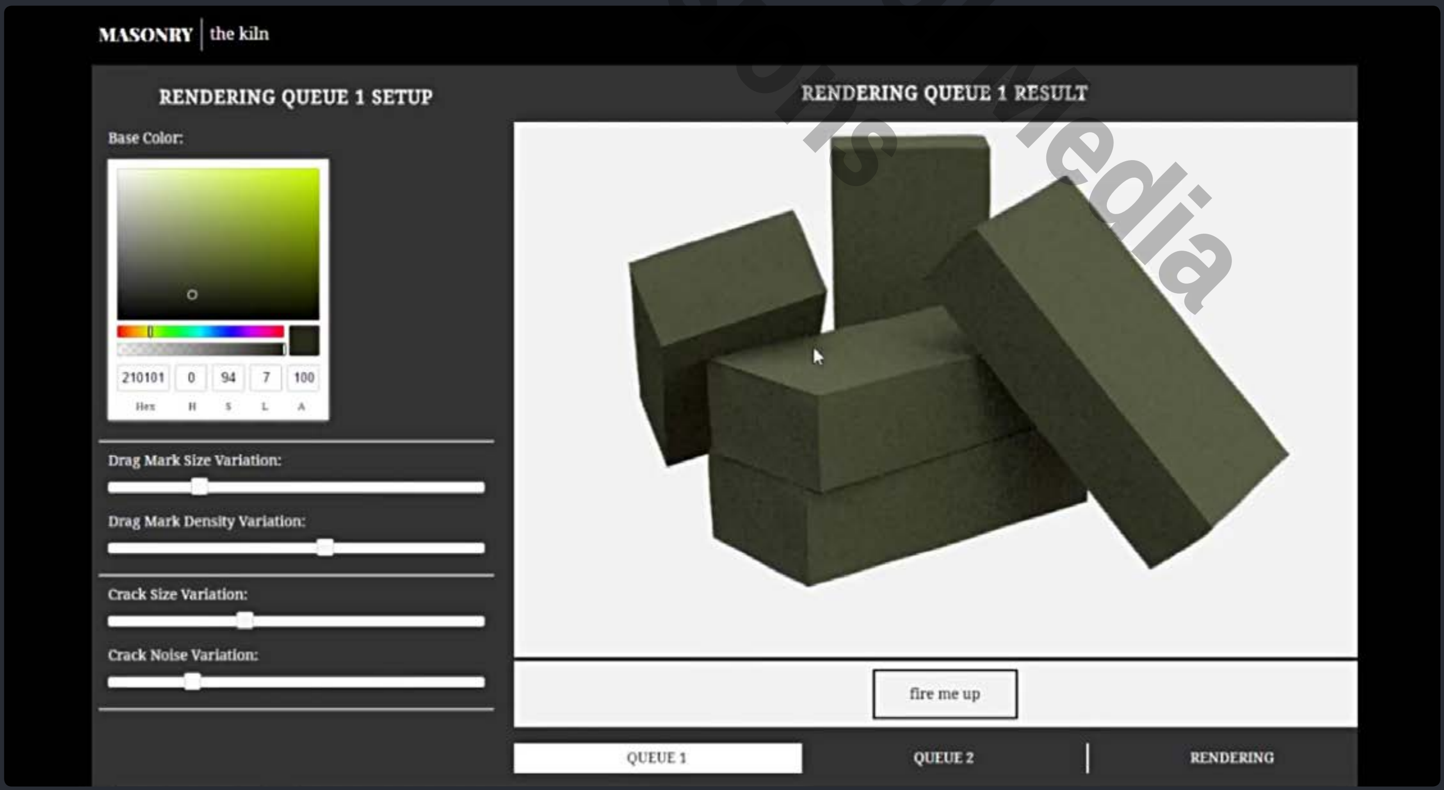
material_final.jpg



laptop_layout_1.jpg

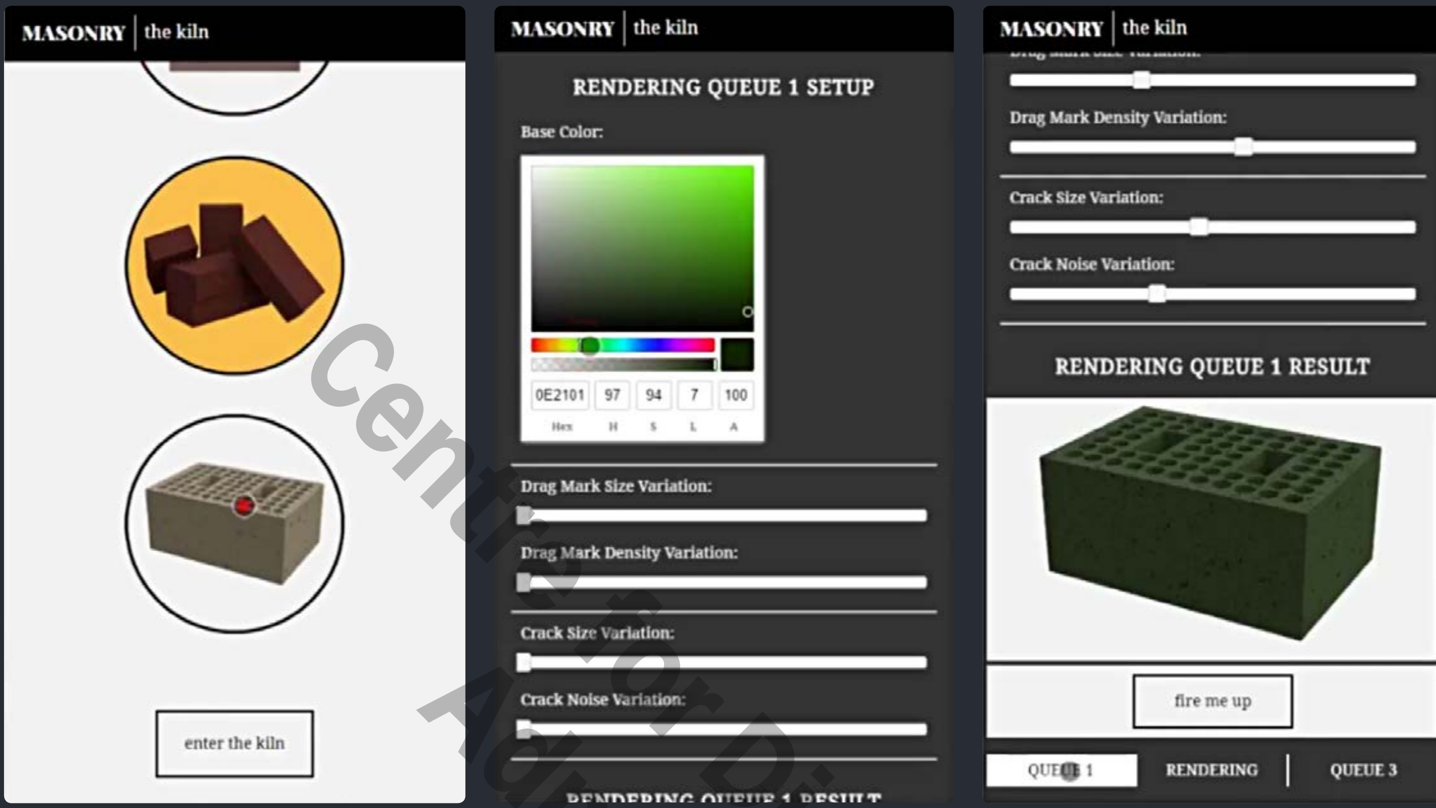


laptop_layout_2.jpg

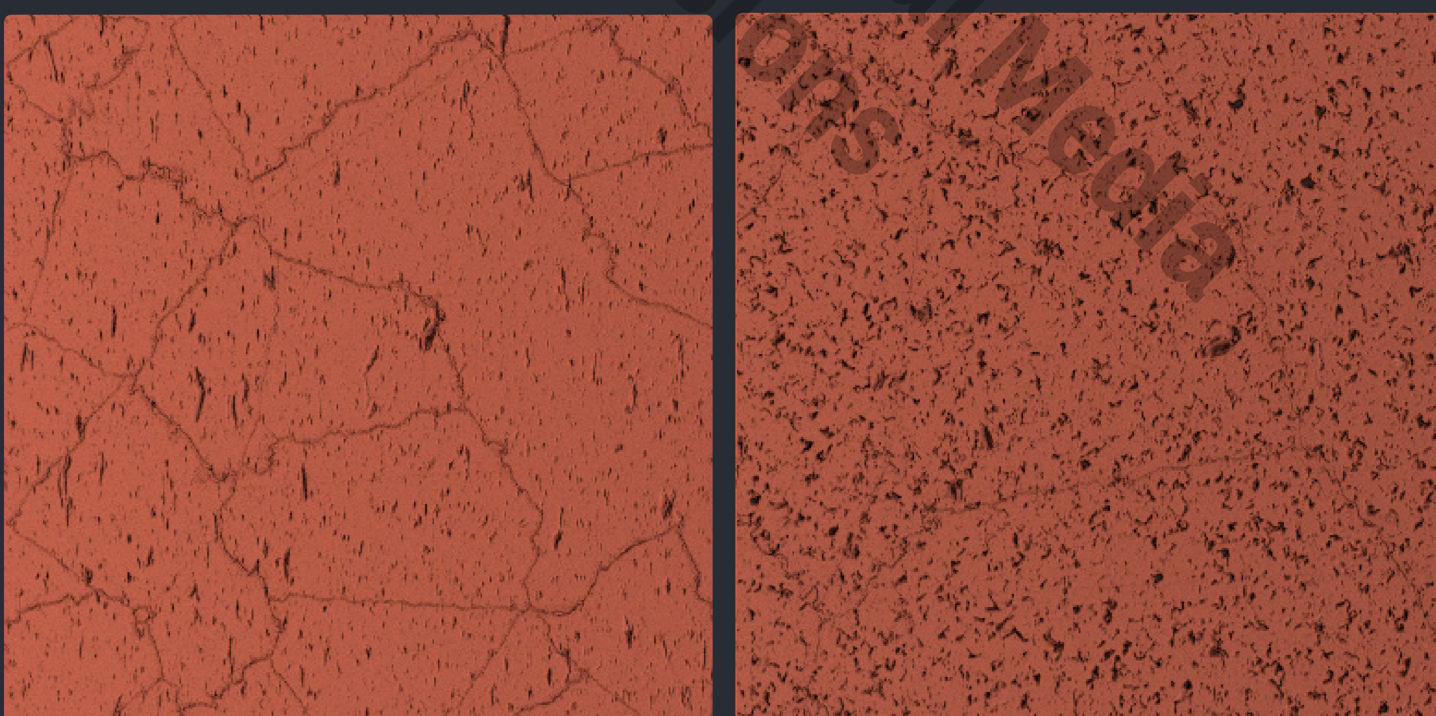


The Kiln - Masonry Texture Generator

mobile_layout.jpg



texture_results.jpg



masonry.py

```
/**
  @file: masonry.py
  @purpose: Parse parameters of the Blender rendering command line and inject them into
the nodes of the cycle material defined in .blend file.
  @author Karl Kim (captainwhale52@gmail.com)
*/

/** Blender rendering command example.
  "-b project/static/blender/masonry1.blend -P project/static/blender/masonry.py -o
      //../render/' + filename + '_#.PNG -x 1 -f 1 -- -hsv_h ' + hue + ' -hsv_s ' +
      saturation + ' -hsv_v ' + value + ' -d_size ' + dsize + ' -d_density ' + ddensity +
      '-c_size ' + csize + ' -c_noise ' + cnoise"
*/

// Try to get parameters from command line
try:
    args = list(reversed(sys.argv))
    idx = args.index("--")
except ValueError:
    params = []
else:
    params = args[:idx][::-1]

// Parse parameters and assign into specific variables
if params[0] == '-hsv_h':
    hsv_h = float(params[1]) / 360.0
if params[2] == '-hsv_s':
    hsv_s = float(params[3]) / 100.0
if params[4] == '-hsv_v':
    hsv_v = float(params[5]) / 100.0
if params[6] == '-d_size':
    d_size = float(params[7]) / 100.0
if params[8] == '-d_density':
    d_density = float(params[9]) / 100.0
if params[10] == '-c_size':
    c_size = float(params[11]) / 100.0
if params[12] == '-c_noise':
    c_noise = float(params[13]) / 100.0

// Find a brick material and assign each value into designated material nodes.
brick = bpy.data.objects["Brick"];
mat = bpy.data.materials["Material"]
nodes = mat.node_tree.nodes
hsv_h_node = nodes.get("hsv_h")
hsv_h_node.outputs[0].default_value = hsv_h
hsv_s_node = nodes.get("hsv_s")
hsv_s_node.outputs[0].default_value = hsv_s
hsv_v_node = nodes.get("hsv_v")
hsv_v_node.outputs[0].default_value = hsv_v
hsv_v_node = nodes.get("d_size")
hsv_v_node.outputs[0].default_value = d_size
... // Continue...
```

app.py

```
/**
  @file: app.py
  @purpose: Simple Flask-based server-side application for getting texture paramteres from
the client, executing Blender with the parameters, and return rendering results back to
the client once rendering processes are completed.
  @author Karl Kim (captainwhale52@gmail.com)
*/

// Handle PUT request of the client and create a new rendering task.
@app.route('/task/<int:task_id>', methods=['PUT'])
def create_task(task_id):
  if not request.json:
    abort(400)
  render(request);
  filename = 'static/render/' + str(request.json['name']) + '_1.PNG'
  return jsonify({'filename': filename})

// Parse the client request and execute a rendering process into a separate thread.
def render(request):
  generic = int(request.json['generic'])
  filename = request.json['name']
  hue = str(request.json['hval'])
  saturation = str(request.json['sval'])
  value = str(request.json['bval'])
  dsize = str(request.json['dsize'])
  ddensity = str(request.json['ddensity'])
  csize = str(request.json['csize'])
  cnoise = str(request.json['cnoise'])

  // Choose different blend file based on the type of texture that the user selects.
  blend = url_for('static', filename='blender/masonry1.blend')
  if generic == 1:
    file = '-b project/static/blender/masonry1.blend -P project/static/blender/masonry.py
        -o //../render/' + filename + '_#.PNG -x 1 -f 1 -- -hsv_h ' + hue + ' -hsv_s '
        + saturation + ' -hsv_v ' + value + ' -d_size ' + dsize + ' -d_density '
        + ddensity + ' -c_size ' + csize + ' -c_noise ' + cnoise
    command = ' '.join([blender, file])
    app.logger.warning(command)     // Print out log message on the server console.
    // Call render process as a separate thread.
    ans = subprocess.call(command, stderr=subprocess.STDOUT, shell=True)

  elif generic == 2:     ...   // Continue...
  elif generic == 3:     ...   // Continue...

if __name__ == '__main__':     // Run a flask server.
  import os
  HOST = os.environ.get('SERVER_HOST', '0.0.0.0')      // Change this value based on server
  try:
    PORT = int(os.environ.get('SERVER_PORT', '9000')) // Change this value based on server
  except ValueError:
    PORT = 80
  app.run(HOST, PORT, threaded=True)
```

task.component.js

```
/**
  @file: task.component.js
  @purpose: React component class to render the UI for changing parameter options, and
rendered results whenever it gets a response from the server.
  @author Karl Kim (captainwhale52@gmail.com)
*/

import React from 'react';
import $ from 'jquery'; // TODO: Change the jquery Ajax funciton into the one of Axios.
import styles from './task.component.css';
import FeaturesComponent from './features.component.js';
import OptionsComponent from './options.component.js';

let TaskComponent = React.createClass({
  // Fetch information of the task
  fetchOptionsOfSelectedFeatures: function(task) {
    let self = this;
    $.ajax({
      url: "/options",
      dataType: 'json',
      cache: false,
      success: function(data) {
        let options = [];
        task.features.forEach(function(feature, i) {
          for (var i = 0; i < data.options.length; i++) {
            if (data.options[i].fid == feature) {
              options.push(data.options[i]);
            }
          }
        });
        self.setState({options: options});
      }.bind(this),
      error: function(xhr, status, err) {
        console.error("/features", status, err.toString());
      }.bind(this);
    });
  },
  componentWillReceiveProps: function(nextProps) {
    this.setState({task: nextProps.task, features: nextProps.features});
    this.fetchOptionsOfSelectedFeatures(nextProps.task);
  },
  // Change rendering options.
  selectFeature(feature) {
    let self = this;
    let index = self.state.task.features.indexOf(feature.id);
    if (index > -1) {
      self.state.task.features.splice(index, 1);
    } else {
      self.state.task.features.push(feature.id);
    }
    $.ajax({
      url: '/tasks/' + self.state.task.id,
```

```
        type: 'PUT',
        contentType: 'application/json',
        data: JSON.stringify(self.state.task),
        dataType: "json",
        success: function(data) {
          self.fetchOptionsOfSelectedFeatures(data.task);
          self.setState({task: data.task});
        }.bind(this),
        error: function(xhr, status, err) {
          console.error("/tasks", status, err.toString());
        }.bind(this)
      });
    },
    // Execute when the "Bake" button is pressed.
    onBake: function(task) {
      let self = this;
      $.ajax({
        url: "/task/" + task.id,
        type: 'PUT',
        contentType: 'application/json',
        data: JSON.stringify(self.state.task),
        dataType: "json",
        success: function(data) {
          self.setState({task: data.task});
        }.bind(this),
        error: function(xhr, status, err) {
          console.error("/tasks", status, err.toString());
        }.bind(this)
      });
    },
    render: function() {  // Rendering UIs under the HTML DOM hierarchy.
      let self = this;
      if (this.state.task) {
        let time = new Date().getTime();
        let progress = {
          width: this.state.task.progress + '%',
        };
        return (
          <div className={styles.wrapper}>
            <div className={styles.progress}>
              <div style={progress}>
                {this.state.task.progress + '%'}
              </div>
            </div>
            <div className={styles.task_wrapper}>
              <div className={styles.task_name}>{this.state.task.name}</div>
              <div className={styles.actions}>
                <div className={styles.reset}>Reset</div>
                <div className={styles.stop}>Stop</div>
                <div className={styles.bake} onClick={()=>
                    {self.onBake(self.state.task);}}>Bake</div>
              </div>
            ...      // Continue...
  module.exports = TaskComponent;
```