IM3080 Design and Innovation Project (AY2021/22 Semester 1)
Individual Report

Name: Vivian Widjaja

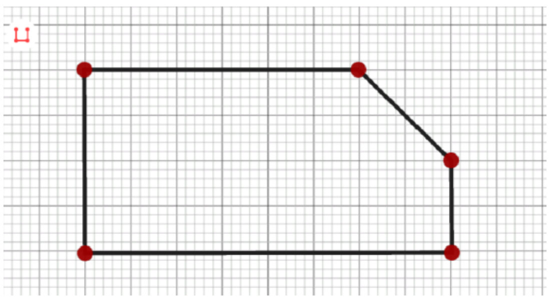
Group No: Group 1

Project Title: IDea – a collaborative home interior design application

Contributions to the Project

1) <u>2D FLOOR PLAN DRAWING</u>

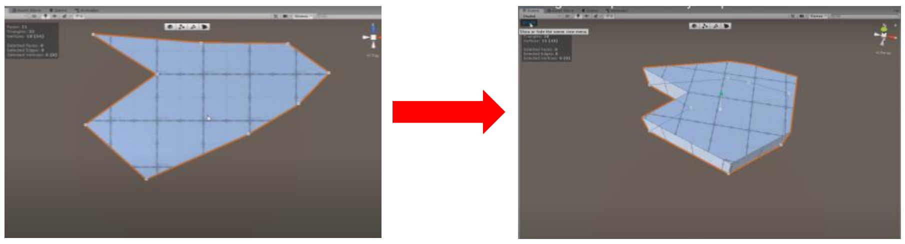
Since the project instructed us to recreate our 1.0 application (Home Design 3D), we intended to implement the 2D floor plan drawing feature into our application. I was tasked to generate 3D house models from 2D floor plan drawings.



This feature allows the user to draw their desired house layout freely on the grid canvas by doing the following:
- Left-click anywhere on the grid to draw the starting point (node) and the next node to produce a line.
- Right-click on the node and line to delete accordingly.
- Move the node around to make adjustments.
- Click the top-left button to loop the line altogether to finalise the floor plan.
- Click the top-left button again to amend the floor plan.

I used Line Prefab with the Line Renderer component in Unity. The source code for this can be found in the Appendix attached.

For the 3D model generation, I planned to do it with one of the following methods with Unity ProBuilder's Poly Shape Tool:
- Assign the 2D floor plan as the top view, then generate 3D walls with a set height.



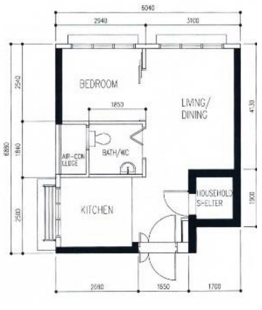
- Instead of generating 2D lines, the user will generate 3D walls straight away when drawing the floor plan. However, the user will view their drawing from the top view so the drawing still looks 2D. After the floor plan is finalised, the view will be adjusted to enable them to see the 3D house.

However, after feedback from the second presentation in Week 6, we decided not to implement this feature to ensure that we will be able to finish the project on time.

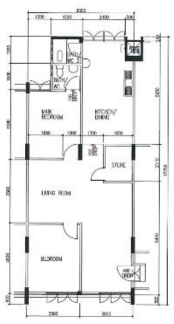
2) <u>3D HOUSE MODELS</u>

Since we are no longer getting the 3D models from 2D floor plans drawn by users, my job is now changed to generate 3D houses from the readily available floor plans. Instead of drawing from scratch, users can choose from the templates provided in the application.

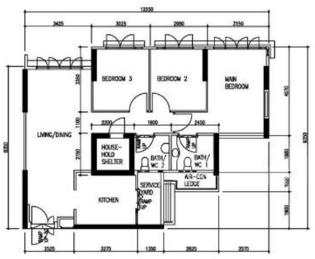
I generated house models from four of the most commonly used HDB floor plans in Singapore:



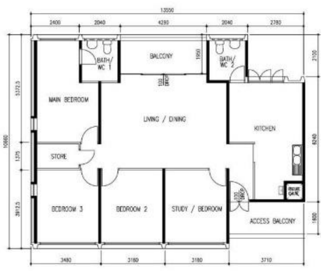
2-room flat



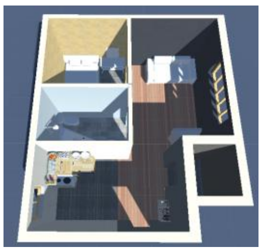
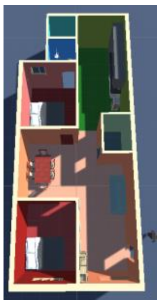
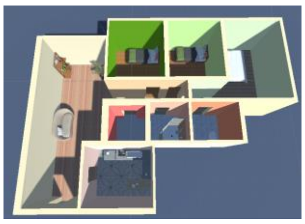
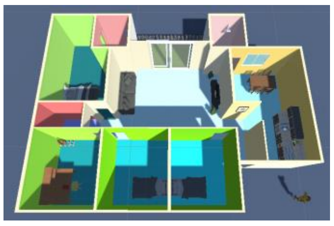
3-room flat

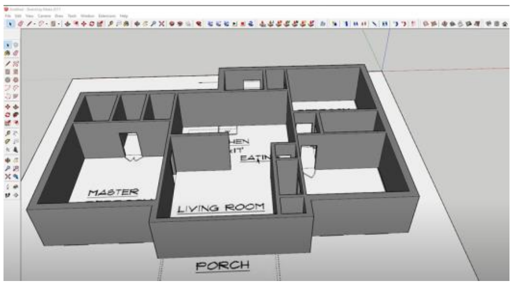


4-room flat



Executive flat



They were created with SketchUp Pro. After drawing the walls one by one as 2D rectangular shapes according to the dimension, I extruded them to build the wall.

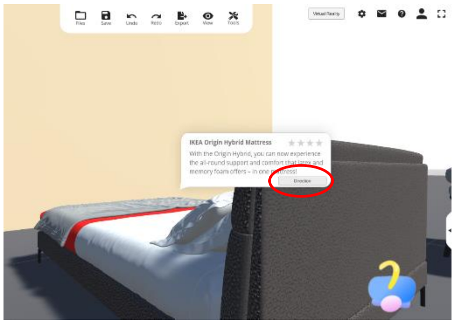
Finally, I added the wallpaper and furniture to the house to make it look vibrant.

I created the house models from scratch instead of importing them. This is because when I tried changing the wallpaper of a single room in Unity for the imported models, the whole house wall material changed too. They are not modular.

Hence, I decided to build the walls one by one in SketchUp to allow the wall and flooring materials, and furniture to be edited individually. This was also to ensure that I can change the models accordingly if they interfere with the other aspects of the project.

3) <u>2D Map</u>

When the user has decided to purchase a certain piece of furniture, they can click on the “Direction” button on the review panel. After clicking on the button, a 2D map will appear, showing the location of the Interior Design (ID) firm.



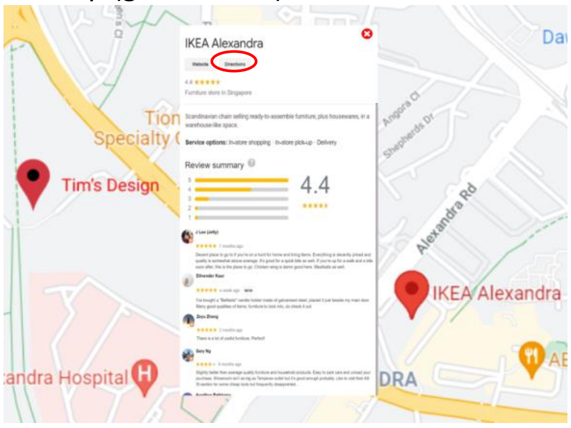
The Anchorage
Alexandra Rd
IKEA Alexandra
ABC Br
ALEXANDRA

The client can also move the map around, zoom in/out. The map will also show other ID firms nearby, in case they want to do a comparison between the different ID firms.

Cafe Colbar
Queenstown EW19
Dawson Place
Tiong Hoe Specialty Coffee
Tim's Design
The Anchorage
Alexandra Rd
IKEA Alexandra
ABC Brickworks
Alexandra Hospital
ALEXANDRA
AYE (Toll road)
+
-

Scripts (on GitHub repositories):
- *CameraMovement.cs*
- *MakeItButton.cs*
- *changeScene2.cs*
- *Popup.cs*

When the ID firm’s float is clicked, the review of the firm will appear. The review may include the furniture’s rating and the ID’s service and capabilities. This will further help the user decide on whether their purchase will be worth their money. The “Direction” button will bring the user to the 3D map (geolocation).



4) <u>Integration to Main Project</u>

Helped to integrate the 3D geolocation and login page functions into the main project.

Connected these aspects altogether with the main scene to ensure that the program flows smoothly from the beginning to the end.

5) <u>Media team</u>

Assisted the team in filming the promotional video and poster creation.

Reflection on Learning Outcome Attainment

Reflect on your experience during your project and the achievements you have relating to <u>at least two</u> of the points below:

(a) Engineering knowledge
(b) Problem Analysis
(c) Investigation
(d) Design/development of Solutions
(e) Modern Tool Usage
(f) The Engineer and Society
(g) Environment and Sustainability
(h) Ethics
(i) Individual and Team Work
(j) Communication
(k) Project Management and Finance
(l) Lifelong Learning

Point 1: <u>(j) Communication</u>

Communication is extremely important especially when this group is made up of 11 people. It was not easy to have communicated smoothly since we met online throughout the whole semester. Despite being divided into 3 subgroups, we still have to communicate with members from other subgroups. Even though it may have seemed that everybody understood what was being said during the main zoom meeting, we could have interpreted the things said differently. I learned to not stick to my first conclusion and always clarify whatever was mentioned during the meeting. This was to ensure that I did not proceed with the project with a misunderstanding.

Point 2: <u>(l) Lifelong Learning</u>

Due to only working with Unity for at most 3 weeks for the intro to design and innovation module before, the platform has a very wide range of features that I had never touched on or even heard before. I was originally assigned to enable users to draw a 2D floor plan, then convert it to a 3D house. To be honest, I didn't know where to start and I had forgotten how to use Unity. I started by investigating how to draw a 2D floor plan in Unity. Even when I managed to find what seemed like a good reference, there were many terms that sounded foreign to me. I decided to spend a substantial amount of time digging through the notes on Unity from last semester and relearn Unity again. I managed to say that I am now confident in navigating the platform and able to understand the technical instructions either from online resources or my groupmates when doing the project.

After doing my investigation, I also discovered that I needed to learn a completely new platform, SketchUp Pro, to create 3D houses for IDea. It was daunting to use a foreign platform, and time was limited to learn something from scratch. There was also no guarantee that the models generated from this platform were able to be exported to Unity properly. I was worried that the other members would not be able to start their part of the project until I had done the main model. Hence, I actually sourced for 3D house models online at first. However, it didn't work out as I have mentioned in the 3D House Models section of this report. In the end, I spent a portion of my time learning the basics, then followed the tutorial to build the models. The models were built on time.

I realized the importance of lifelong learning and it is worth it to take your time to learn something new or even relearn something. I can say that I know how to use 2 new applications from working on this project.

Appendix

```
using System.Collections;
using System;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class DotController : MonoBehaviour, IDragHandler, IPointerClickHandler
{
   public LineController line;
   public int index;

   public Action<DotController> OnDragEvent;

   public void OnDrag(PointerEventData eventData)
   {
      OnDragEvent?.Invoke(this);
   }

   public Action<DotController> OnRightClickEvent;
   public Action<DotController> OnLeftClickEvent;
   public void OnPointerClick(PointerEventData eventData)
   {
      if (eventData.pointerId == -2)
      {
         //Right Click

         OnRightClickEvent?.Invoke(this);
      }
      else if (eventData.pointerId == -1)
      {
         //Left Click
         OnLeftClickEvent?.Invoke(this);
      }
   }

   public void SetLine(LineController line)
   {
      this.line = line;
   }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LineController : MonoBehaviour
{
    private LineRenderer lr;
    private List<DotController> dots;

    private void Awake()
    {
        lr = GetComponent<LineRenderer>();
        lr.positionCount = 0;
        dots = new List<DotController>();
    }

    public DotController GetFirstPoint()
    {
        return dots[0];
    }
    public void AddPoint(DotController dot)
    {
        dot.index = dots.Count;
        dot.SetLine(this);

        lr.positionCount++;
        dots.Add(dot);
    }

    public void SplitPointsAtIndex(int index, out List<DotController> beforeDots, out List<DotController>
afterDots)
    {
        List<DotController> before = new List<DotController>();
        List<DotController> after = new List<DotController>();

        int i = 0;
        for (; i < index; i++)
        {
            before.Add(dots[i]);
        }
        i++;
        for (; i < dots.Count; i++)
        {
            after.Add(dots[i]);
        }

        beforeDots = before;
        afterDots = after;

        dots.RemoveAt(index);
    }

    public void ToggleLoop()
    {
        lr.loop = !lr.loop;
    }
    public bool isLooped()
    {
        return lr.loop;
    }
    private void LateUpdate()
    {
        if (dots.Count >= 2)
        {
            for(int i = 0; i < dots.Count; i++)
            {
                lr.SetPosition(i, dots[i].transform.position);
            }
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

public class PenCanvas : MonoBehaviour, IPointerClickHandler
{
    public Action OnPenCanvasLeftClickEvent;
    public Action OnPenCanvasRightClickEvent;

    public void OnPointerClick(PointerEventData eventData)
    {
        if (eventData.pointerId == -1)
        {
            OnPenCanvasLeftClickEvent?.Invoke();
        }
    }
}
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PenTool : MonoBehaviour
{
    [Header("Pen Canvas")]
    [SerializeField] private PenCanvas penCanvas;

    [Header("Dots")]
    [SerializeField] Transform dotParent;
    [SerializeField] private GameObject dotPrefab;

    [Header("Lines")]
    [SerializeField] Transform lineParent;
    [SerializeField] private GameObject linePrefab;
    private LineController currentLine;

    [Header("Loop Toggle")]
    [SerializeField] Image loopToggle;
    [SerializeField] Sprite loopSprite;
    [SerializeField] Sprite unloopSprite;

    private void Start()
    {
        penCanvas.OnPenCanvasLeftClickEvent += AddDot;
        penCanvas.OnPenCanvasRightClickEvent += EndCurrentLine;
    }

    public void ToggleLoop()
    {
        if (currentLine != null)
        {
            currentLine.ToggleLoop();
            loopToggle.sprite = (currentLine.isLooped()) ? unloopSprite : loopSprite;
        }
    }



    private void EndCurrentLine()
    {
        if(currentLine != null)
        {
            loopToggle.enabled = false;
            currentLine = null;
        }
    }

    private void AddDot()
    {
        if (currentLine == null)
        {
            currentLine = Instantiate(linePrefab, Vector3.zero, Quaternion.identity,
lineParent).GetComponent<LineController>();
        }

        DotController dot = Instantiate(dotPrefab, GetMousePosition(), Quaternion.identity,
dotParent).GetComponent<DotController>();
        dot.OnDragEvent += MoveDot;
        dot.OnRightClickEvent += RemoveDot;
        dot.OnLeftClickEvent += SetCurrentLine;



        currentLine.AddPoint(dot);
    }

    private void SetCurrentLine(DotController dot)
    {
        EndCurrentLine();
        currentLine = dot.line;

        loopToggle.enabled = true;
        loopToggle.sprite = (currentLine.isLooped()) ? unloopSprite : loopSprite;
    }
    private void RemoveDot(DotController dot)
    {
        LineController line = dot.line;
        line.SplitPointsAtIndex(dot.index, out List<DotController> before, out List<DotController> after);
        Destroy(line.gameObject);
        Destroy(dot.gameObject);

        LineController beforeLine = Instantiate(linePrefab, Vector3.zero, Quaternion.identity,
lineParent).GetComponent<LineController>();
        for (int i = 0; i < before.Count; i++)
        {
            beforeLine.AddPoint(before[i]);
        }

        LineController afterLine = Instantiate(linePrefab, Vector3.zero, Quaternion.identity,
lineParent).GetComponent<LineController>();
        for (int i = 0; i < after.Count; i++)
        {
            afterLine.AddPoint(after[i]);
        }
    }

    private void MoveDot(DotController dot)
    {
        dot.transform.position = GetMousePosition();
    }

    private Vector3 GetMousePosition()
    {
        Vector3 worldMousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        worldMousePosition.z = 0;

        return worldMousePosition;
    }
}
```