




# Examples of some cool HCI projects

by Pooya Khaloo

November 21<sup>th</sup> 2019

University of Massachusetts Amherst

- 
- Human-Computer Interaction (HCI) is the study of **how people use computers** throughout their lives.
  - HCI research seeks to develop user interfaces that are **useful, usable, and enjoyable**.
  - It focuses on activities ranging from design to development to evaluation of computer systems, with a goal of **understanding how computers and technology affect people and society**.



# Outline

- My masters project – **Code Park**
- My PhD project – **CommunityClick**
- Math Boxes
- Electrick
- LumiWatch



# Purpose of showcasing Code Park

- How you can improve a project through an **iterative design** process?
- How to design a **user study** to evaluate your system?



```
['l', 'o', 'u', 'y', NOR_e, NOR_o, NOR_a)
```

```

    *list, short randval);
    list = *list, short rem);
    *f);

    char v[];
    list = *list, short rem, short vc, char vow[]);
    : *list);

    char *argv[]

    // Not in accordance with UNIX philosophy, but is not prioritized since this is not specified in the assignment
    r, "vs failed. Incorrect number of arguments!\n\nUSAGE: %vs <print|random|replace|remove|len> <file>\n", argv[0], argv[0]);

```

```
name, 'p0')) == NULL) {  
    err = "cannot find file: %s", name);
```

```

otherload;
init(f);

am, "print")) {
    otherload, 0);

mp(param, "random")) {
    se(1000));
}

```

**SAMSUNG**

90 Watt AC/DC Universal Notebook Charger



# Introduction

[illegible]



# Problem

- IDEs have been around for decades and they are all inherently 2D
  - Writing code
  - Learning new language
  - Understanding a new codebase
- Not an issue for expert users



BUT

How about the novice users?



## 8 Learning new Codebase:

---

01

Uses  
memory

02

Has  
cognitive  
load

03

It is boring



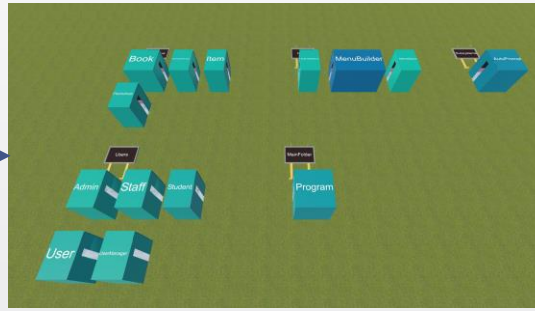
# Code Visualizers

- Assist developers in gaining insights into a codebase
- Make learning and understanding code easier

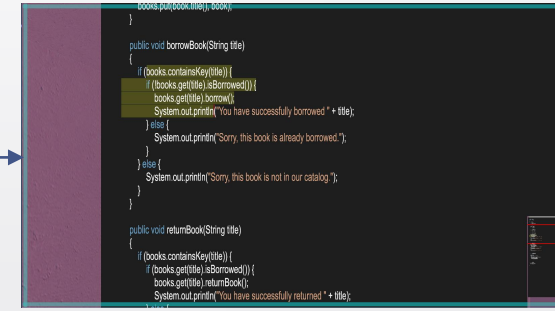
# Design Process



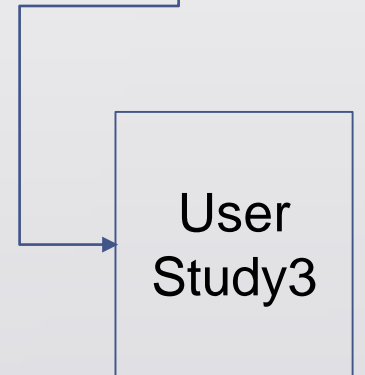
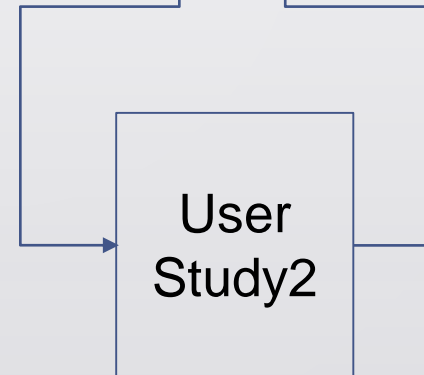
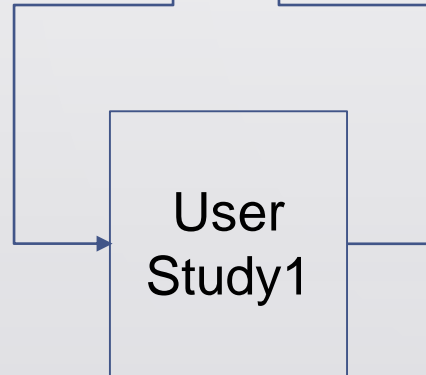
Code House



Code Park: 3D Visualization Tool



Code Park V2: 3D IDE







# Code House



- Each class is attached to wall as a wallpaper
- Each wallpaper is syntax aware
- User explicitly define location of each wallpaper
- User can freely walk in the house environment



The House is fixed model

Each room has different color and furniture

It helps user remember where they are and location of each wallpaper





Why we started with house?

Can the users remember where they placed **class X** in this code-filled house, similar to the way they remember where they left their **toothbrush** in real life?



# Navigation

- Mini Map
- Follow Path
- Follow Path, automated
- Teleport

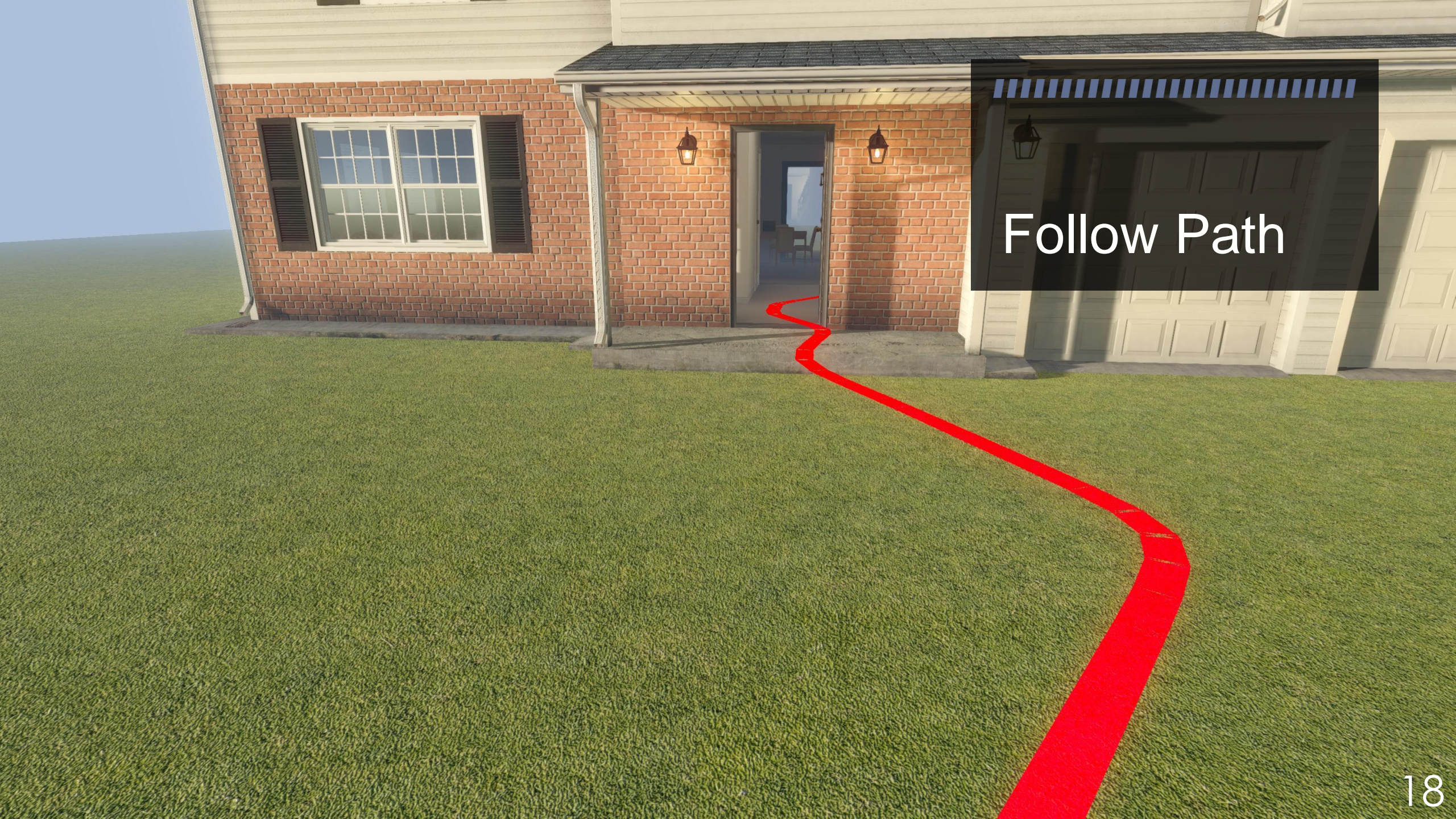


Help:  
E: Open Editor  
R: On wall editing  
F1: Cancel editing  
F2: Cancel dropdown  
F5: Compile  
1: Teleport  
2: Path navigator  
3: Map navigator  
Q: Auto path follower

# Mini Map







Follow Path





# Teleport

```
using CoffeeAuth.Models;
using Microsoft.Maker.RemoteWiring;
using Microsoft.Maker.Serial;
using SQLitePCL;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Windows.UI.ViewManagement;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=234238.

namespace CoffeeAuth
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to
    /// </summary>
    public sealed partial class MainPage : Page
    {
        User lastUser;
    }
}
```



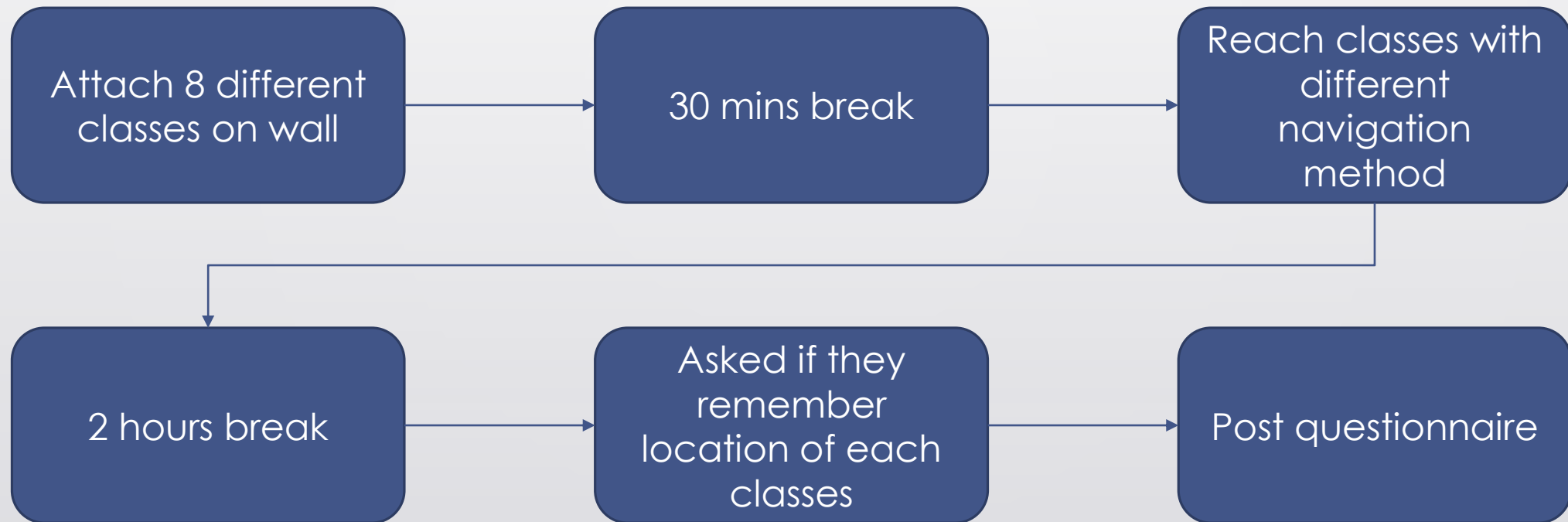
# Code House Evaluation



Can users remember where they put their specific code in the house and which navigation methods is better to help them remembering?

# User Study

- 5 participants (all male ranging in age from 22 to 28)



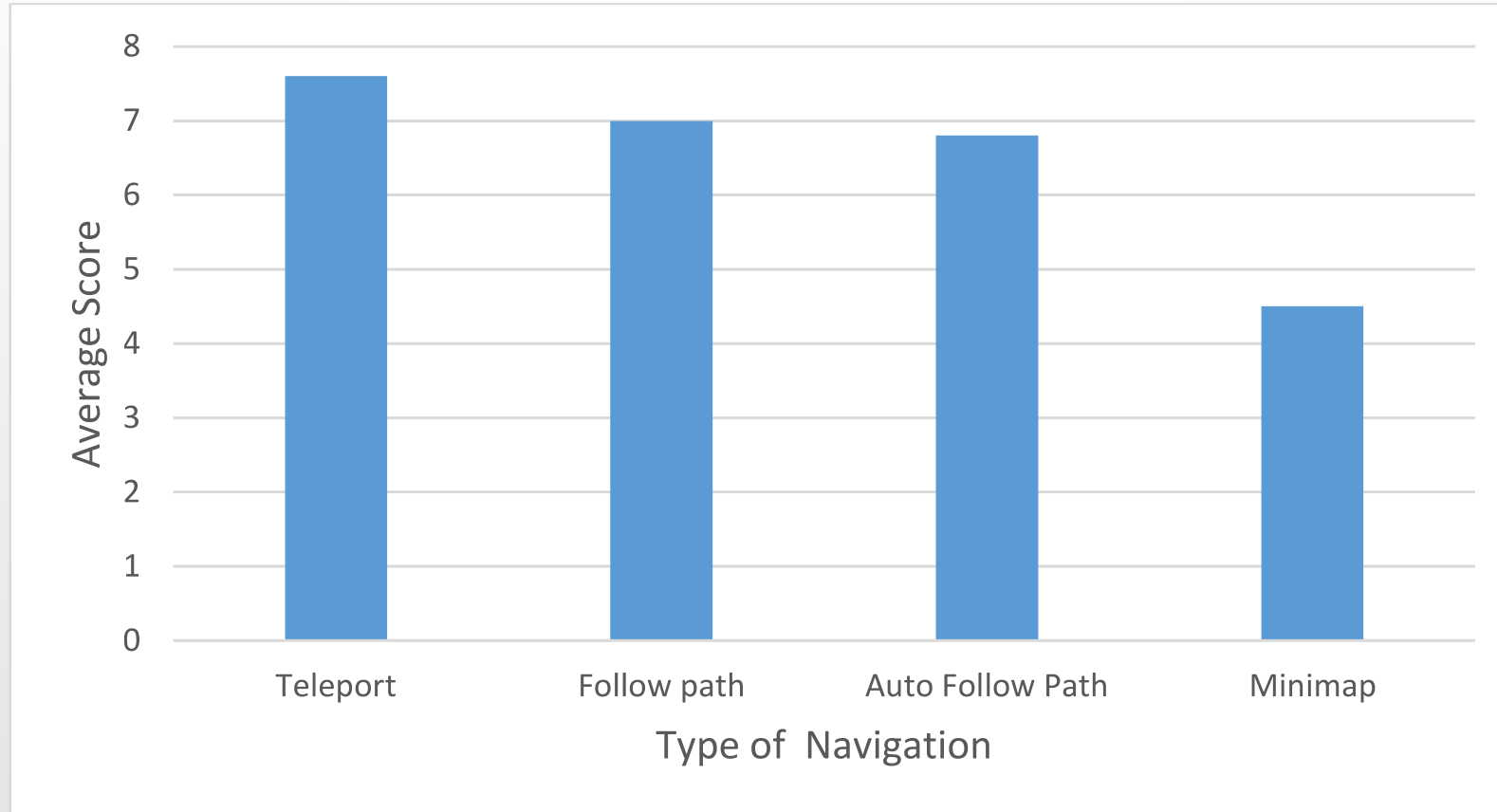




## Result

- 3/5 remembered the location of all 8 classes
- Other 2 forgot location of classes they reached with **Teleport**
- They scored each navigation method between 1 - 10





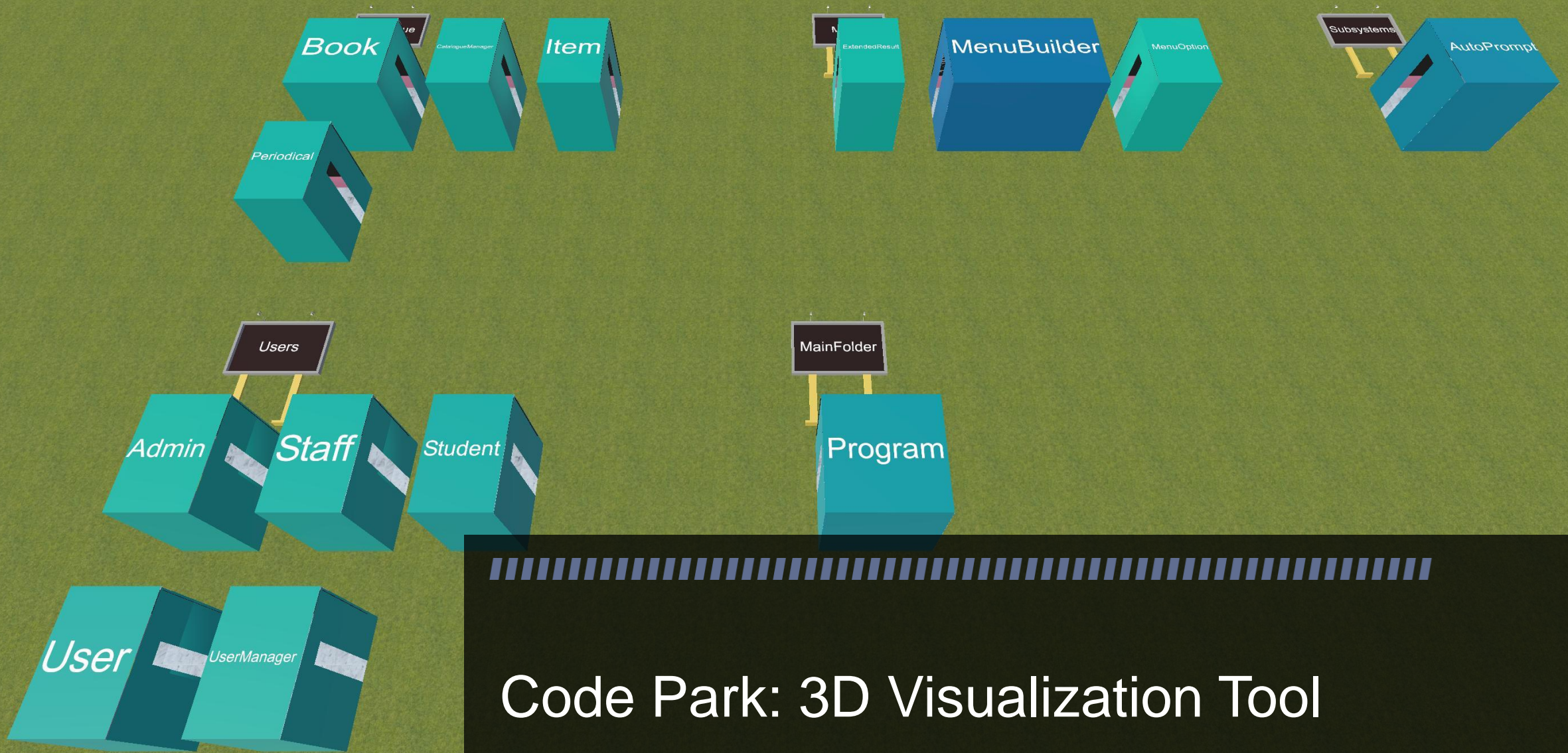
Post questionnaire result



# Discussion

- Positive answer to our main question
- Appealing and fun (Based on participants' opinions)
- Not easy to use
- Not expandable
- Leads to next version ...







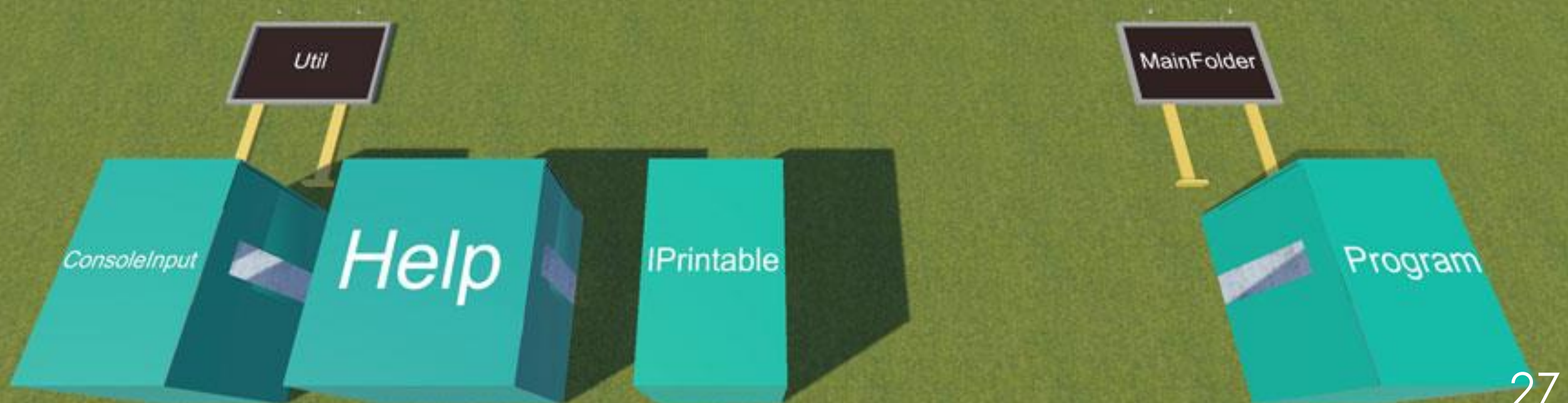
This Version supports C#

Rooms create automatically

Each room represents a class

Files in a same directory resulted in adjacent rooms

Size & color of each room represent size of the class

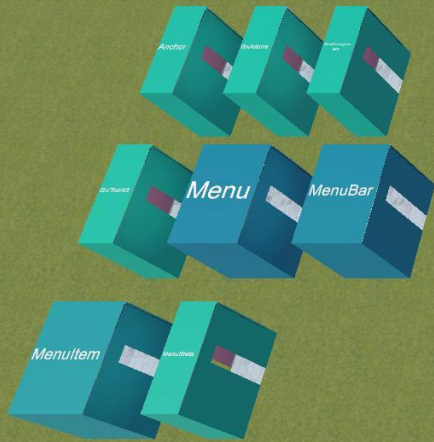
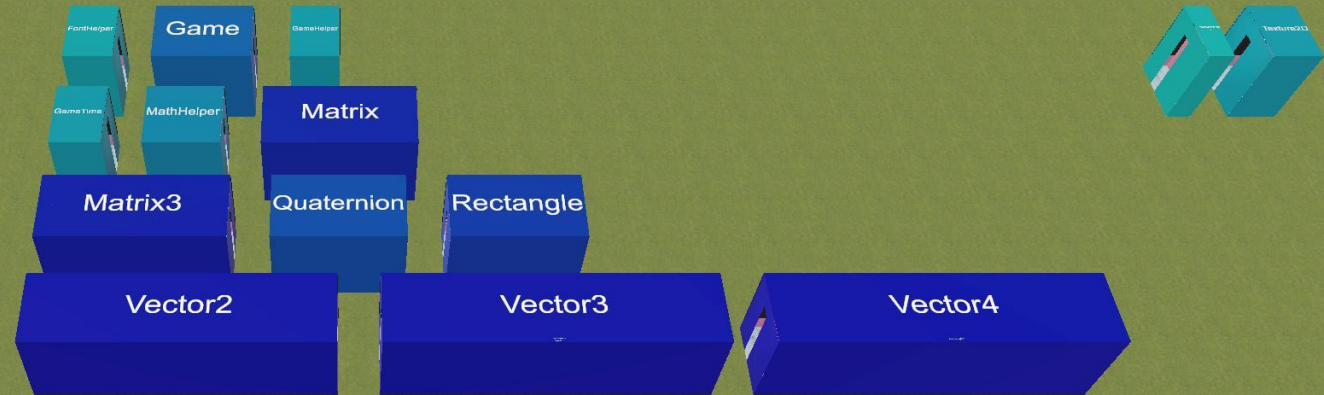




Code Park supports different views



# God View



Users can see all the classes

Users can choose each class for more details

Transfer user to **Player View** by selecting each room



# Player View

CardFace

Users can walk freely between classes like FPS games

Users can see codes in detail

Transfer user to **Code View** by clicking on code canvas

```
[Serializable]
public class GameEngine
{
```

```
    const int maxLevel = 3;
    const int point = 2;
    public CardBack[,] CardBack { get; set; }
    public CardFace[,] CardFace { get; set; }
    public Player Player { get; set; }
    public List<CardPosition> Check { get; set; }
    public int Level { get; set; }
```

```
    public GameEngine(CardBack[,] cardBack, CardFace[,] cardFace, Player player, int level)
    {
        this.CardFace = cardFace;
        this.CardBack = cardBack;
        this.Player = player;
        this.Check = new List<CardPosition>();
        this.Level = level;
    }
}
```

## Code View

Mini Map to show where user is in code view

Background and syntax color resample Microsoft Visual Studio







All transition between **God View**, **Player View** and **Code View** is animated to maintain the users' sense of spatial awareness

```

public CardFace(string cardName)
public override void DrawSelf(int row, int col)
public void Print(CardPosition topLeft)
public override bool Equals(object obj)
public override int GetHashCode()
public static void PrintCardOne(CardPosition top)
public static void PrintCardTwo(CardPosition top)
public static void PrintCardThree(CardPosition top)
public static void PrintCardFour(CardPosition top)
public static void PrintCardFive(CardPosition top)
public static void PrintCardSix(CardPosition top)
public static void PrintCardSeven(CardPosition top)
public static void PrintCardEight(CardPosition top)
public static void PrintCardNine(CardPosition top)

```

CardFace

Use syntax parsing to get list of methods

Show these methods both on top of classes  
in God View

And on wall in the room in Player View

```

...ck[,] cardBack, CardFace[,] cardFace, Player player)
Data(SerializationInfo info, StreamingContext context)
public void Run()
private void CheckForEquals()
private void CheckForNextLevel()
private void PrintPlayerInfo(string message)
... (ConsoleColor consoleColor, int dimensionOne, int dimensionTwo)
... ter(CardPosition position, ConsoleColor consoleColor)
private void Grid()

```



```
using System;
using System.Collections.Generic;
using System.Threading;
using System.Linq;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
```

```
namespace MemoryGames
{
    [Serializable]
    public class GameEngine
    {
        const int maxLevel = 3;
        const int point = 2;
        public CardBack[,] CardBack { get; set; }
        public CardFace[,] CardFace { get; set; }
        public Player Player { get; set; }
        public List<CardPosition> Check { get; set; }
        public int Level { get; set; }

        public GameEngine(CardBack[,] cardBack, CardFace[,] cardFace, Player player, int level)
        {
            this.CardFace = cardFace;
            this.CardBack = cardBack;
            this.Player = player;
            this.Check = new List<CardPosition>();
            this.Level = level;
        }

        public void GetObjectData(SerializationInfo info, StreamingContext context)
        {
            info.AddValue("cardBack", this.CardBack);
            info.AddValue("cardFace", this.CardFace);
            info.AddValue("player", this.Player);
            info.AddValue("check", this.Check);
            info.AddValue("level", this.Level);
        }

        public void Run()
        {
            int dimensionZero = 8;
            int dimensionOne = 8;

            Refresh();
            Grid();
            while (true)
            {
                PrintPlayerInfo();
                PointerVisualisation(ConsoleColor.Black, dimensionZero, dimensionOne);
                if (Console.KeyAvailable)
                {
                    var keyInfo = Console.ReadKey(true);
                    if (keyInfo.Key.Equals(ConsoleKey.UpArrow))
                    {
                        if (dimensionZero > 0)
                        {
                            dimensionZero--;
                        }
                        if (keyInfo.Key.Equals(ConsoleKey.DownArrow))
                        {
                            dimensionZero++;
                        }
                        if (keyInfo.Key.Equals(ConsoleKey.LeftArrow))
                        {
                            if (dimensionOne > 0)
                            {
                                dimensionOne--;
                            }
                        }
                        if (keyInfo.Key.Equals(ConsoleKey.RightArrow))
                        {
                            dimensionOne++;
                        }
                        if (keyInfo.Key.Equals(ConsoleKey.Spacebar))
                        {
                            if ([CardFace(dimensionZero, dimensionOne)].IsVisible)
                            {
                                Check.Add(new CardPosition(dimensionZero, dimensionOne));
                            }
                        }
                        if (keyInfo.Key.Equals(ConsoleKey.Escape))
                        {
                            foreach (var position in this.Check)
                            {
                                this.CardFace[position.X, position.Y].IsVisible = false;
                                this.CardBack[position.X, position.Y].IsVisible = true;
                            }
                            GameManager.SaveGame(this.CardBack, this.CardFace,
                                this.Player, this.Check, this.Level);
                            GameManager.IsGame();
                        }
                        dimensionZero = CardFace.GetLength(0);
                        dimensionOne = CardFace.GetLength(1);
                        PointerVisualisation(ConsoleColor.Yellow, dimensionZero, dimensionOne);
                        if (Check.Count == 1)
                        {
                            CheckForEquals();
                        }
                        Thread.Sleep(200);
                    }
                }
            }
        }

        private void CheckForEquals()
        {
            if (Check.Count == 1)
            {
                CardPosition firstCard = Check[0];
                this.CardFace[firstCard.X, firstCard.Y].IsVisible = true;
                this.CardBack[firstCard.X, firstCard.Y].IsVisible = false;
            }
            if (Check.Count == 2)
            {
                CardPosition firstCard = Check[0];

```

```
                this.CardFace[secondCard.X, secondCard.Y].IsVisible = true;
                this.CardBack[secondCard.X, secondCard.Y].IsVisible = false;
                Refresh();
                if (this.CardFace[firstCard.X, firstCard.Y] !=
                    this.CardFace[secondCard.X, secondCard.Y])
                {
                    this.CardFace[firstCard.X, firstCard.Y].IsVisible = false;
                    this.CardBack[firstCard.X, firstCard.Y].IsVisible = true;
                    this.CardFace[secondCard.X, secondCard.Y].IsVisible = false;
                    this.CardBack[secondCard.X, secondCard.Y].IsVisible = true;
                    this.Player.SuccessCoefficient--;
                    PrintPlayerInfo();
                }
                else
                {
                    this.Player.Score += this.Player.SuccessCoefficient > 0 ?
                        point * this.Player.SuccessCoefficient : point;
                    PrintPlayerInfo("Success");
                    if (CheckForGameEnd())
                    {
                        if (this.Level < maxLevel)
                        {
                            NextLevel();
                            GameManager.ClearBackground();
                            Refresh();
                            Grid();
                        }
                        else
                        {
                            GameManager.ClearBackground();
                            GameManager.SaveGame(this.Player); //saved from run
                            GameManager.WinGame();
                        }
                    }
                }
                Check.Clear();
                Thread.Sleep(100);
                Refresh();
            }

            private void NextLevel()
            {
                this.Level++;
                this.CardBack = CardRandomPosition.GetCardBackLevel();
                this.CardFace = CardRandomPosition.GetCardFaceLevel(this.CardBack.GetLength(0), this.CardBack.GetLength(1));
            }

            private bool CheckForGameEnd()
            {
                int sumOfVisibleCard = 0;
                foreach (var element in CardFace)
                {
                    if (element.IsVisible == true)
                    {
                        sumOfVisibleCard++;
                    }
                }
                if (sumOfVisibleCard ==
                    CardFace.GetLength(0) * CardFace.GetLength(1))
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }

            private void PrintPlayerInfo(string message)
            {
                const int half = 2;
                Console.SetCursorPosition(Console.WindowWidth / half - this.Player.Name.Length / half, 0);
                Console.WriteLine(this.Player.Name);
                Console.SetCursorPosition(Console.WindowWidth / half - this.Player.Score.ToString().Length
                    half, 10);
                Console.WriteLine("Score: {0}", this.Player.Score);
                Console.SetCursorPosition(Console.WindowWidth / half - message.Length / half, 12);
                Console.WriteLine(message);
                Console.ResetColor();
            }

            private void PointerVisualisation(ConsoleColor consoleColor, int dimensionZero, int dimensionOne)
            {
                CardPosition position = CardPosition.GeneratePosition(dimensionZero, dimensionOne);
                PrintPointer(position, consoleColor);
            }

            private void PrintPointer(CardPosition position, ConsoleColor consoleColor)
            {
                Console.SetCursorPosition(position.X - 1, position.Y - 1);
                Console.BackgroundColor = consoleColor;
                Console.WriteLine(new string('-', 10));
                for (int i = 0; i < 9; i++)
                {
                    position.X++;
                    Console.SetCursorPosition(position.X - 1, position.Y - 1);
                    Console.WriteLine("");
                    Console.SetCursorPosition(position.X - 1 + 9, position.Y - 1);
                    Console.WriteLine("");
                }
                Console.SetCursorPosition(position.X - 1, position.Y - 1);
                Console.WriteLine(new string('-', 10));
                Console.ResetColor();
            }

            void Refresh()
            {
                for (int row = 0; row < this.CardBack.GetLength(0); row++)
                {
                    for (int col = 0; col < this.CardBack.GetLength(1); col++)

```

```
                }
            }

            private void Grid()
            {
                const int gridRightWhiteSpace = 2;
                const int gridDownWhiteSpace = 1;
                Console.ForegroundColor = ConsoleColor.Red;

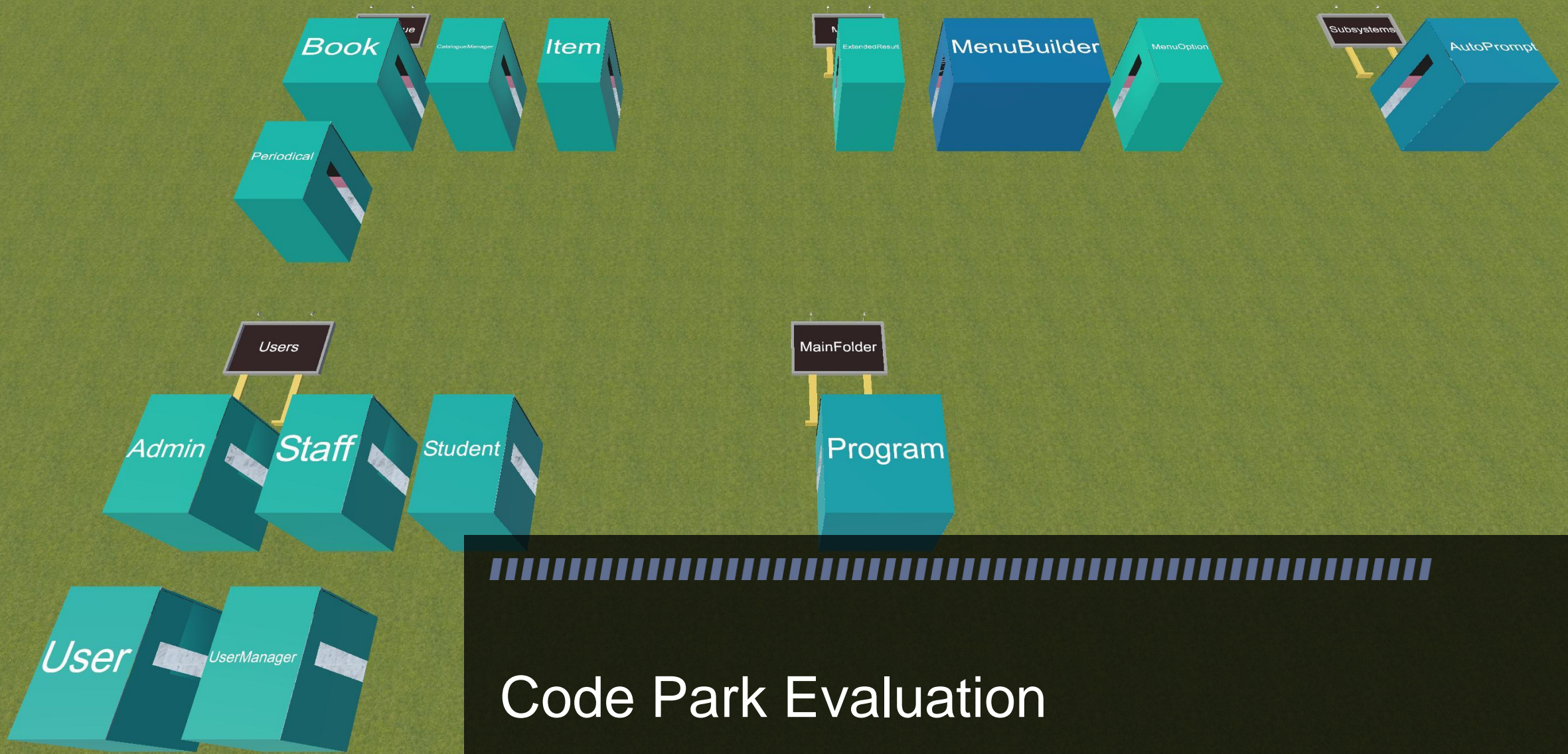
                for (int element = 0; element < this.CardBack.GetLength(1) - 1; element++)
                {
                    int col = this.CardBack[0, element].TopLeft.X + this.CardBack[0, element].Width + gridRightWhiteSpace;
                    int startOfGrid = this.CardBack[0, element].TopLeft.Y - this.CardBack[0, element].Width;
                    int endOfGrid = this.CardBack[this.CardBack.GetLength(0) - 1, element].TopLeft.Y;
                    for (int row = startOfGrid; row < endOfGrid; row++)
                    {
                        Console.SetCursorPosition(col, row);
                        Console.ForegroundColor = ConsoleColor.Red;
                        Console.WriteLine("");
                    }
                }
                for (int element = 0; element < this.CardBack.GetLength(0) - 1; element++)
                {
                    int row = this.CardBack[element, 0].TopLeft.Y + gridDownWhiteSpace;
                    int startOfGrid = this.CardBack[element, 0].TopLeft.X;
                    int endOfGrid = this.CardBack[element, this.CardBack.GetLength(1) - 1].TopLeft.X + this.CardBack[element,
                        this.CardBack.GetLength(1) - 1].Width;
                    for (int col = startOfGrid; col < endOfGrid; col++)
                    {
                        Console.SetCursorPosition(col, row);
                        Console.ForegroundColor = ConsoleColor.Red;
                        Console.WriteLine("");
                    }
                }
            }
        }
    }
}
```

With syntax parsing we also add Go To Definition

Quickly transfer to the location inside a file where a user-defined type or a variable is defined for the first time

This transfer is also animated









# Code Park Goals

- Ease of use
- Help with code understanding
- Being fun and engaging

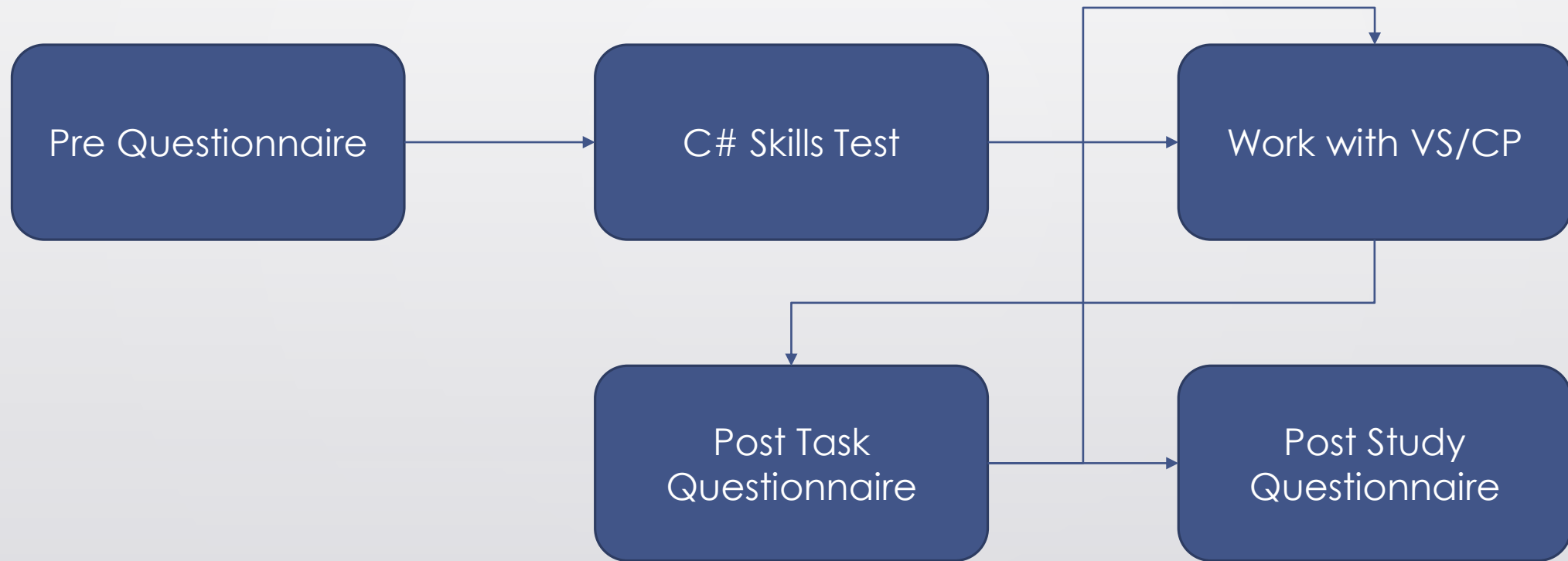




# Comparing Visual Studio with Code Park

# User Study

- 28 participants (22 males and 6 female ranging in age from 18 to 31 with a mean age of 22.8)



# User Study

- To avoid learning effect we used two different codebases
- Therefore we had four groups:

Group Name	First Tool	First Code Base	Second Tool	Second Code Base
A	Visual Studio	LM	Code Park	MG
B	Code Park	MG	Visual Studio	LM
C	Visual Studio	MG	Code Park	LM
D	Code Park	LM	Visual Studio	MG





# Tasks

- T1: Find a valid username to login into the program.
- T2: Find an abstract class in the code base.
- T3: Determine the relationship between classes A and B.
- T4: Find a designed bug that causes a program crash.
- T5: Pinpoint a reasonable location in the code for adding the necessary logic to support feature X.

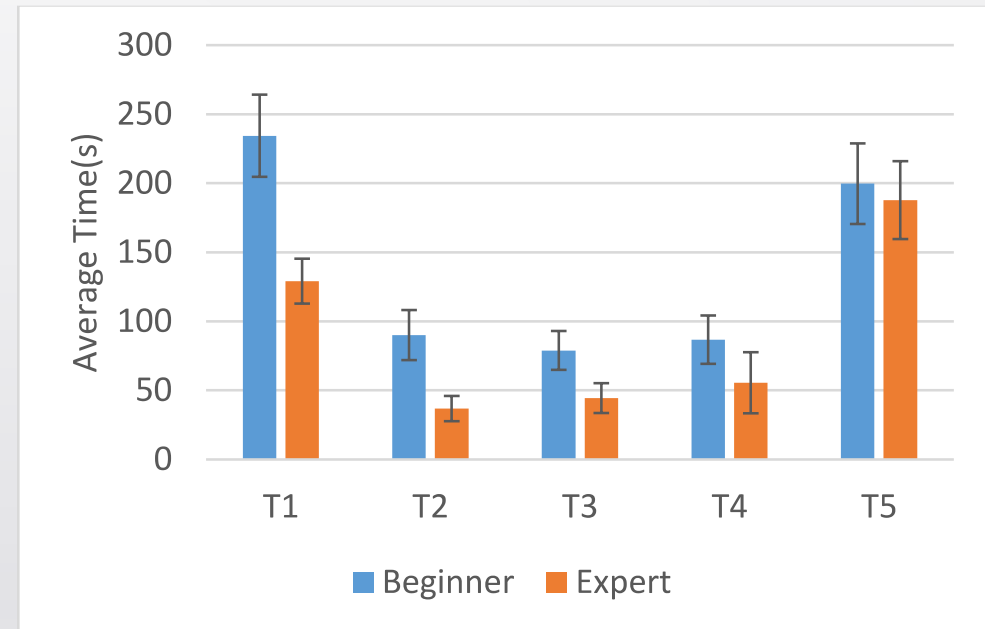


# Metric

- Quantitative:
  - Each task was timed
- Qualitative:
  - Post Task Questionnaire (Likert scale 1 - 7)
  - Post Study Questionnaire (Choose between VS/CH)

## Result (Quantitative)

- Beginners took more time to finish their tasks

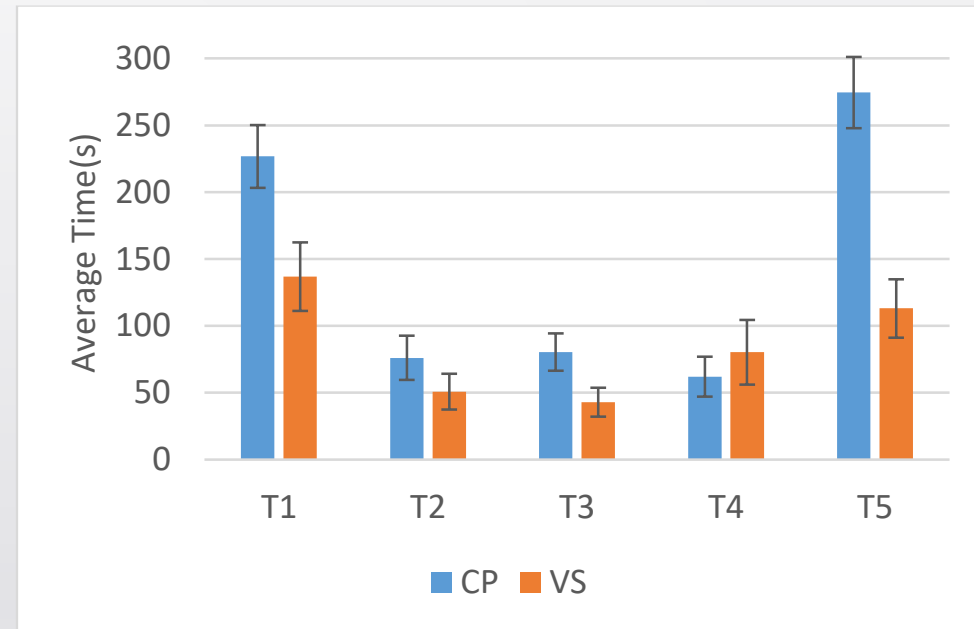


Mean time to task completion by experience level



## Result (Quantitative)

- Task took more time to complete with Code Park
- Code Park has animation
- 3D interaction is slower
- First time using Code Park

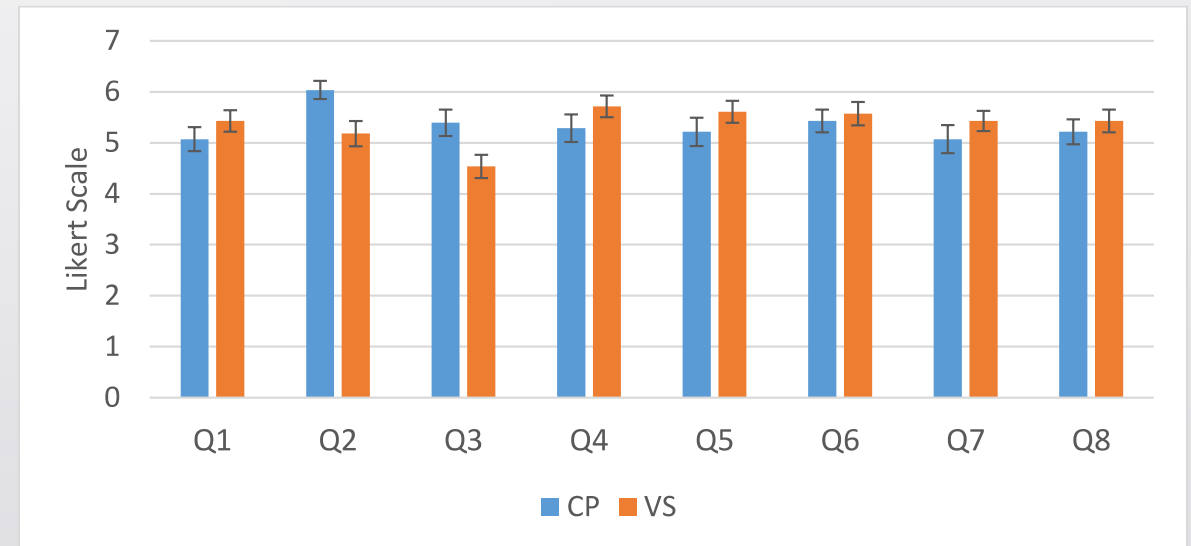


Mean time to task completion based on the tool used

# Result (Qualitative)

## Post Task Questionnaire

- Q1** I found it easy to work with Code Park/Visual Studio.
- Q2** I found it easy to become familiar with Code Park/Visual Studio.
- Q3** Code Park/Visual Studio helps me become familiar with code base's structure.
- Q4** It was easy to navigate through the code with Code Park/Visual Studio.
- Q5** It was easy to find the definition of some variable with Code Park/Visual Studio.
- Q6** How much did you like Code Park/Visual Studio?
- Q7** How did you feel when using the tool?
- Q8** It was easy to find what I wanted in the code using Code Park/Visual Studio.



Mean responses to the post task questionnaire for Code Park and Visual Studio

# Result (Qualitative)

Post Study Questionnaire	
SQ1	Which tool is more comfortable to use?
SQ2	Which tool is more likable?
SQ3	Which tool is more natural?
SQ4	Which tool is easier to use?
SQ5	Which tool is more fun to use?
SQ6	Which tool is more frustrating?
SQ7	Which tool helps you more in remembering the code base structure?
SQ8	Which tool do you prefer for learning a code base?
SQ9	Which tool do you prefer for a code base you are already familiar with for additional work?
SQ10	Which tool do you prefer for finding a particular class/variable?
SQ11	Which tool do you prefer for tracking down a bug?
SQ12	Overall, which tool is better?

Question	Visual Studio	Code Park
SQ1	21	7
SQ2	8	20
SQ3	16	12
SQ4	17	11
SQ5	0	28
SQ6	11	13
SQ7	5	23
SQ8	7	21
SQ9	19	9
SQ10	15	13
SQ11	19	9
SQ12	16	11

All participants unanimously agreed that Code Park is **fun** to use





- Participant 1: *“I enjoyed a lot as it was the first time I was viewing the code in 3D environment.”*
- Participant 6: *“It is much easier to understand the overall code structure.”*
- Participant 17: *“It is very friendly and easy.”*
- Participant 19: *“We could easily access classes and methods.”*
- Participant 26: *“The use of spatial representation was well used.”*



# Discussion

- We achieved our goals
  - Ease of use
  - Help with code understanding
  - Being fun and engaging
- Users ask for two important features
  - Editing code
  - Compiling code
- Leads to next version ...

```

books.put(book.title(), book);
}

public void borrowBook(String title)
{
    if (books.containsKey(title)) {
        if (!books.get(title).isBorrowed()) {
            books.get(title).borrow();
            System.out.println("You have successfully borrowed " + title);
        } else {
            System.out.println("Sorry, this book is already borrowed.");
        }
    } else {
        System.out.println("Sorry, this book is not in our catalog.");
    }
}
}

```

```

public void returnBook(String title)
{
    if (books.containsKey(title)) {
        if (books.get(title).isBorrowed()) {
            books.get(title).returnBook();
            System.out.println("You have successfully returned " + title);
        } else {

```

Code Park: 3D IDE



Add support for Java

Users can create new project

Users can add classes, replace and delete them

Users can continue working on their project later





```

books.put(book.title(), book);
}

public void borrowBook(String title)
{
    if (books.containsKey(title)) {
        if (!books.get(title).isBorrowed()) {
            books.get(title).borrow();
            System.out.println("You have successfully borrowed " + title);
        } else {
            System.out.println("Sorry, this book is already borrowed");
        }
    } else {
        System.out.println("Sorry, this book is not in our catalog.");
    }
}

public void returnBook(String title)
{
    if (books.containsKey(title)) {
        if (books.get(title).isBorrowed()) {
            books.get(title).returnBook();
            System.out.println("You have successfully returned " + title);
        } else {
            System.out.println("Sorry, this book is not borrowed");
        }
    }
}

```

On Code View users can edit code

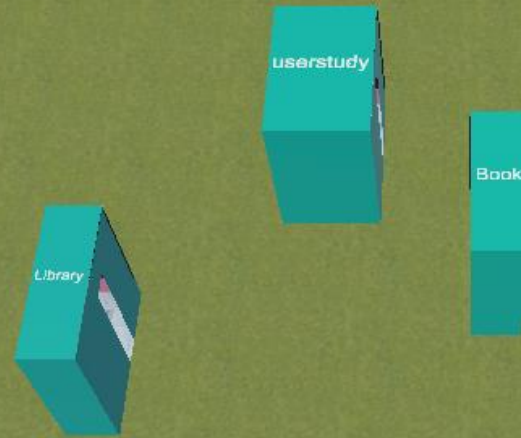
Blue border indicates editing mode

Editing code supports all the basic functionality (select, copy, cut, paste)





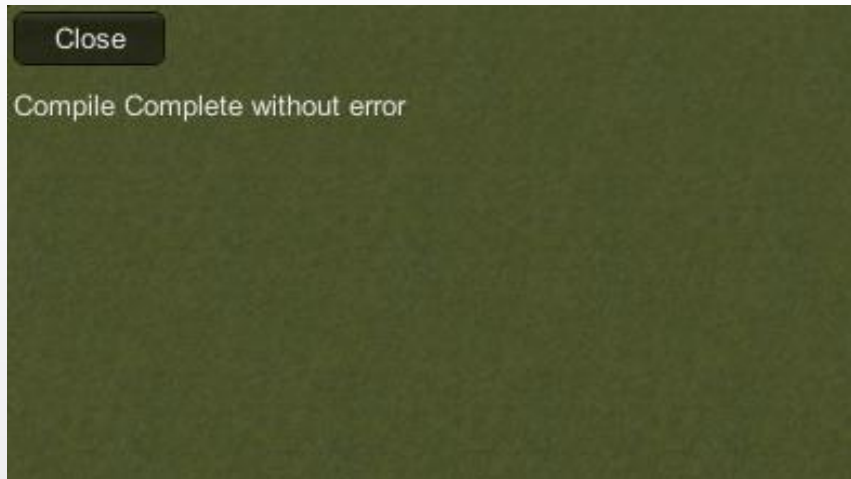
# Terminal



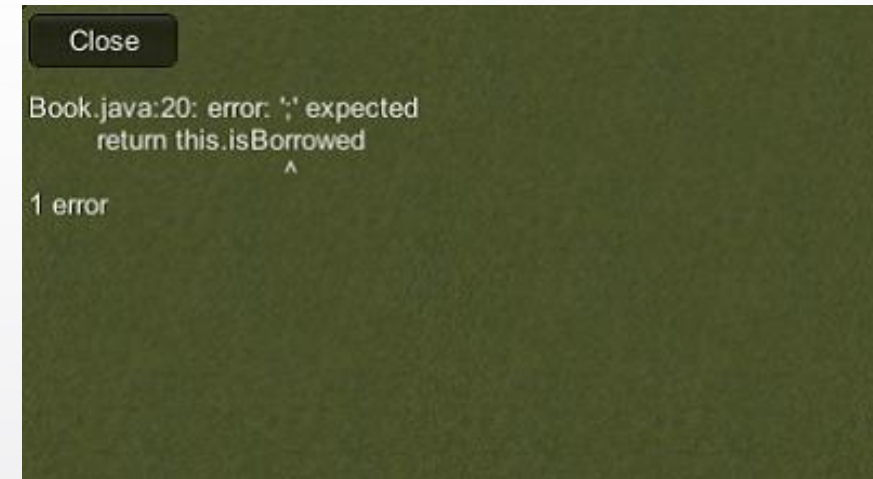
Close

```
Book.java:20: error: ';' expected
return this.isBorrowed
                ^
1 error
```

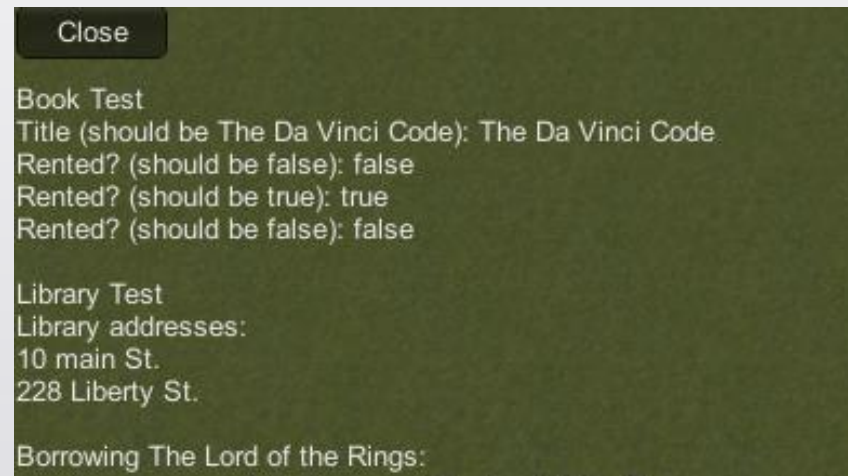




Compiled without error



Compiled with errors



Compiled program's output

```
public class Book
{
    private String title;
    private boolean isBorrowed;
```

',' expected

```
    this.isBorrowed = false;
}

public String title()
{
    return this.title;
}

public boolean isBorrowed()
{
    return this.isBorrowed;
}

public void borrow()
{
```

Indicates error location on code

Hovering red dots shows more details of error

```

books.put(book.title(), book);
}

public void borrowBook(String title)
{
    if (books.containsKey(title)) {
        if (!books.get(title).isBorrowed()) {
            books.get(title).borrow();
            System.out.println("You have successfully borrowed " + title);
        } else {
            System.out.println("Sorry, this book is already borrowed.");
        }
    } else {
        System.out.println("Sorry, this book is not in our catalog.");
    }
}
}

```

```

public void returnBook(String title)
{
    if (books.containsKey(title)) {
        if (books.get(title).isBorrowed()) {
            books.get(title).returnBook();
            System.out.println("You have successfully returned " + title);
        } else {

```

## Code Park V2 Evaluation



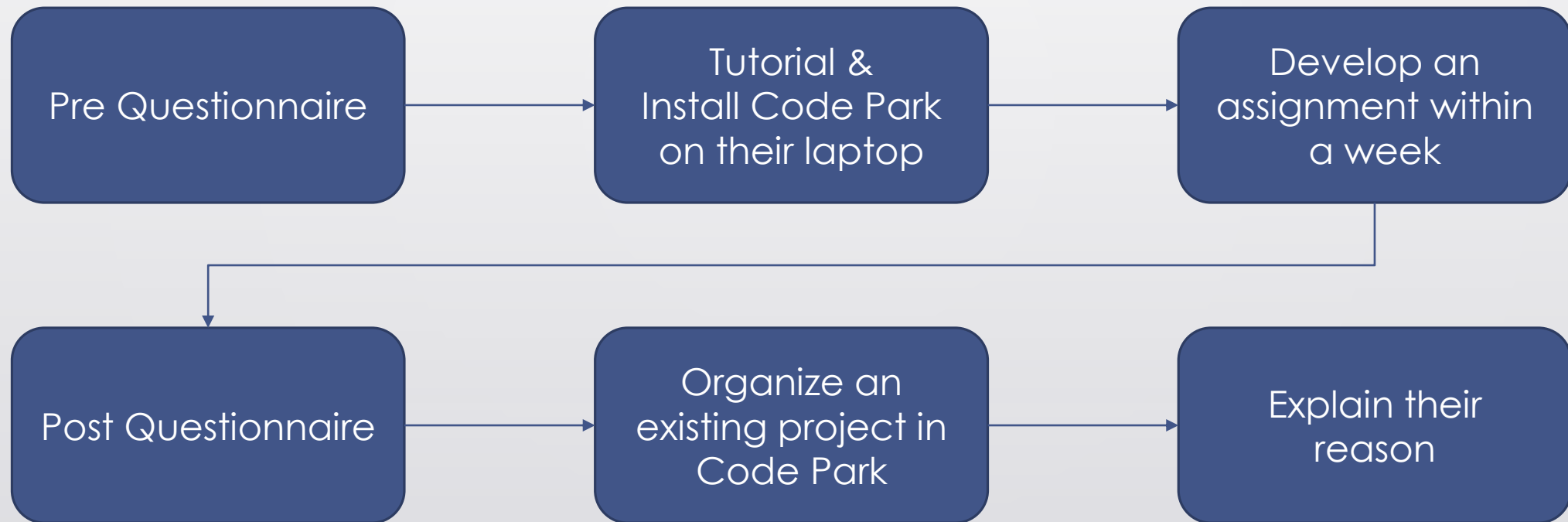


# User Study Goal

- Evaluate our two new features
- Evaluate usability of this version

# User Study

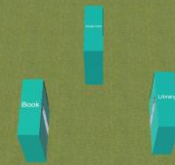
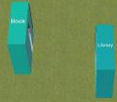
- 9 participants (8 males and 1 female ranging in age from 18 to 29 with a mean age of 22.8)



All the participants finished the assignment

Some of them finished with three classes and organize them as a triangle shape

Other finished it with only two classes and organize them in a line

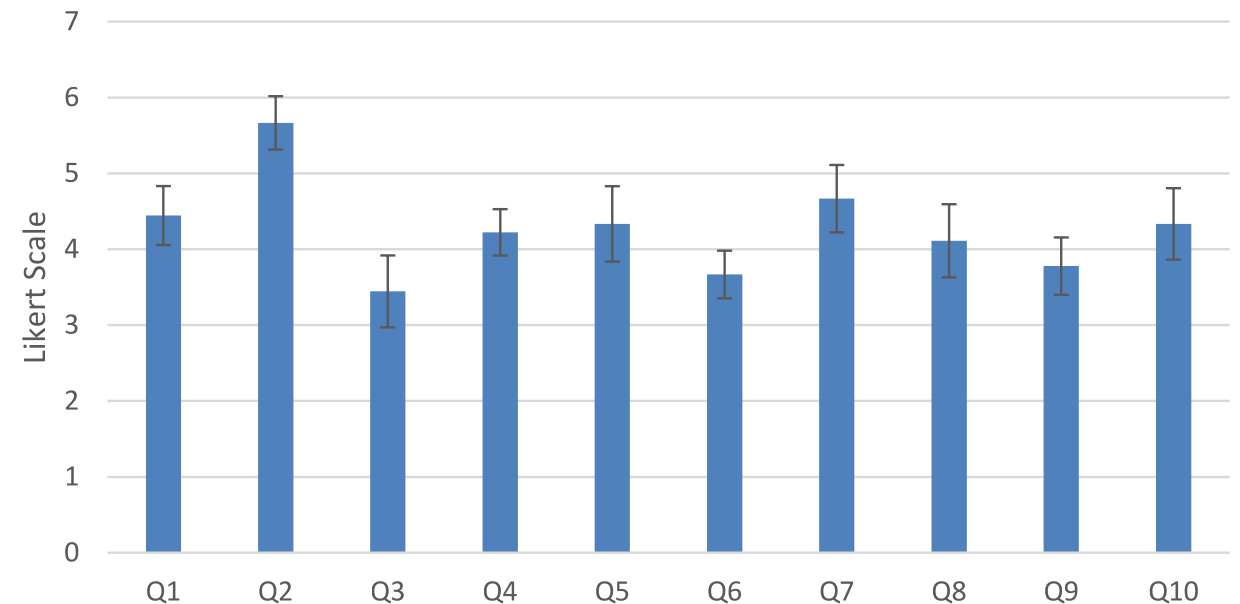




## Post Study Questionnaire

- Q1** I found it easy how to use Code Park.
- Q2** I found it easy how to learn using Code Park.
- Q3** I think that I would like to use Code Park frequently.
- Q4** I found the various functions in Code Park were well integrated.
- Q5** It was easy to navigate through the code with Code Park.
- Q6** It was easy to write code in Code Park.
- Q7** It was easy to work on a project with Code Park.
- Q8** How much did you like the Code Park?
- Q9** How did you feel when using the interface?
- Q10** It was easy to find what I wanted in the code using Code Park.

- Except Q3, Q6 and Q9 all the other answers are beyond the average scale
- The results are similar to previous study
- Easy to learn and use Code Park (Q1 & Q2)
- Easy to work on the project (Q7)
- All features integrated well in Code Park (Q4)





# Organizing Existing Project

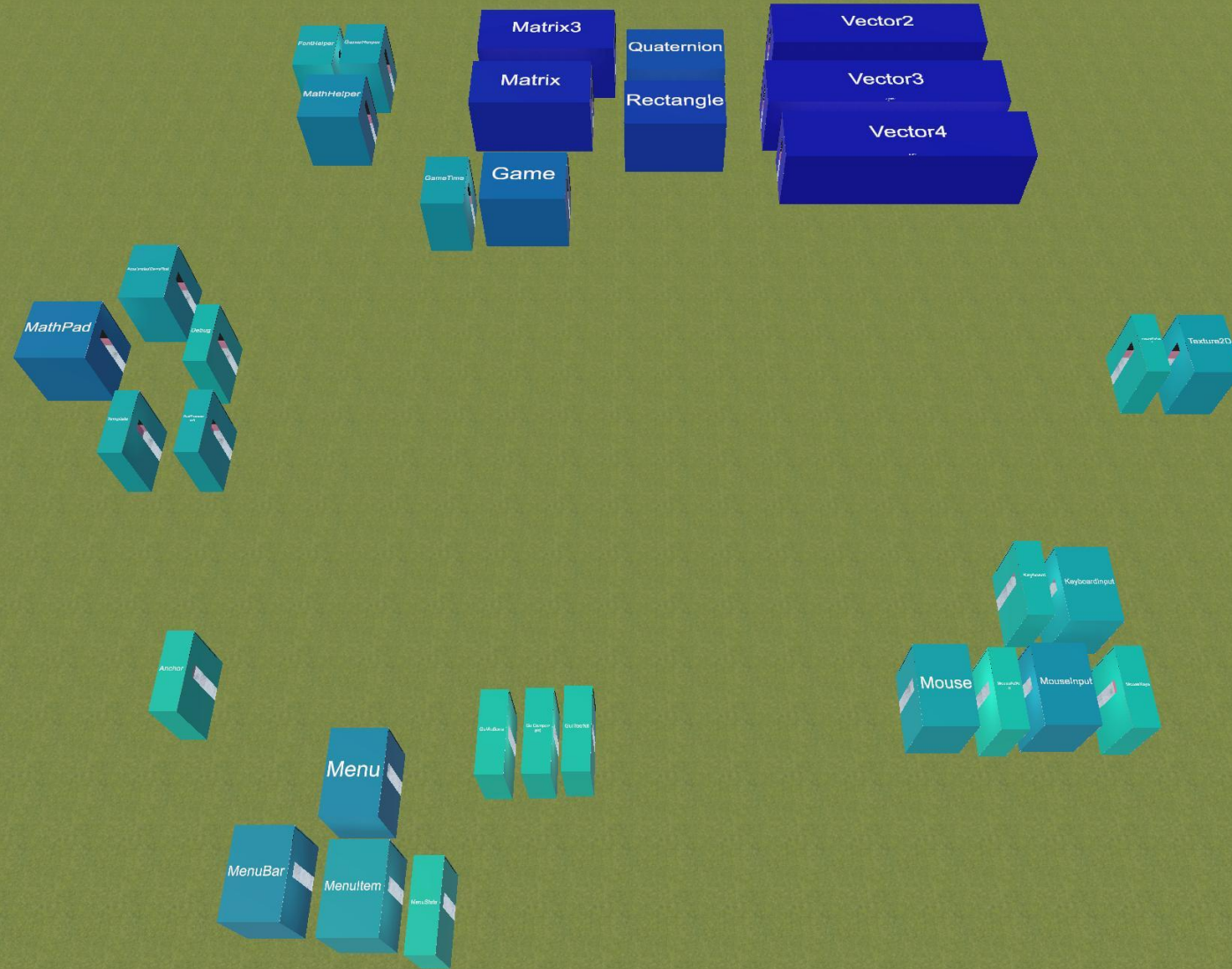
- The project contains 33 classes
- Divided participants into two groups



## Group 1

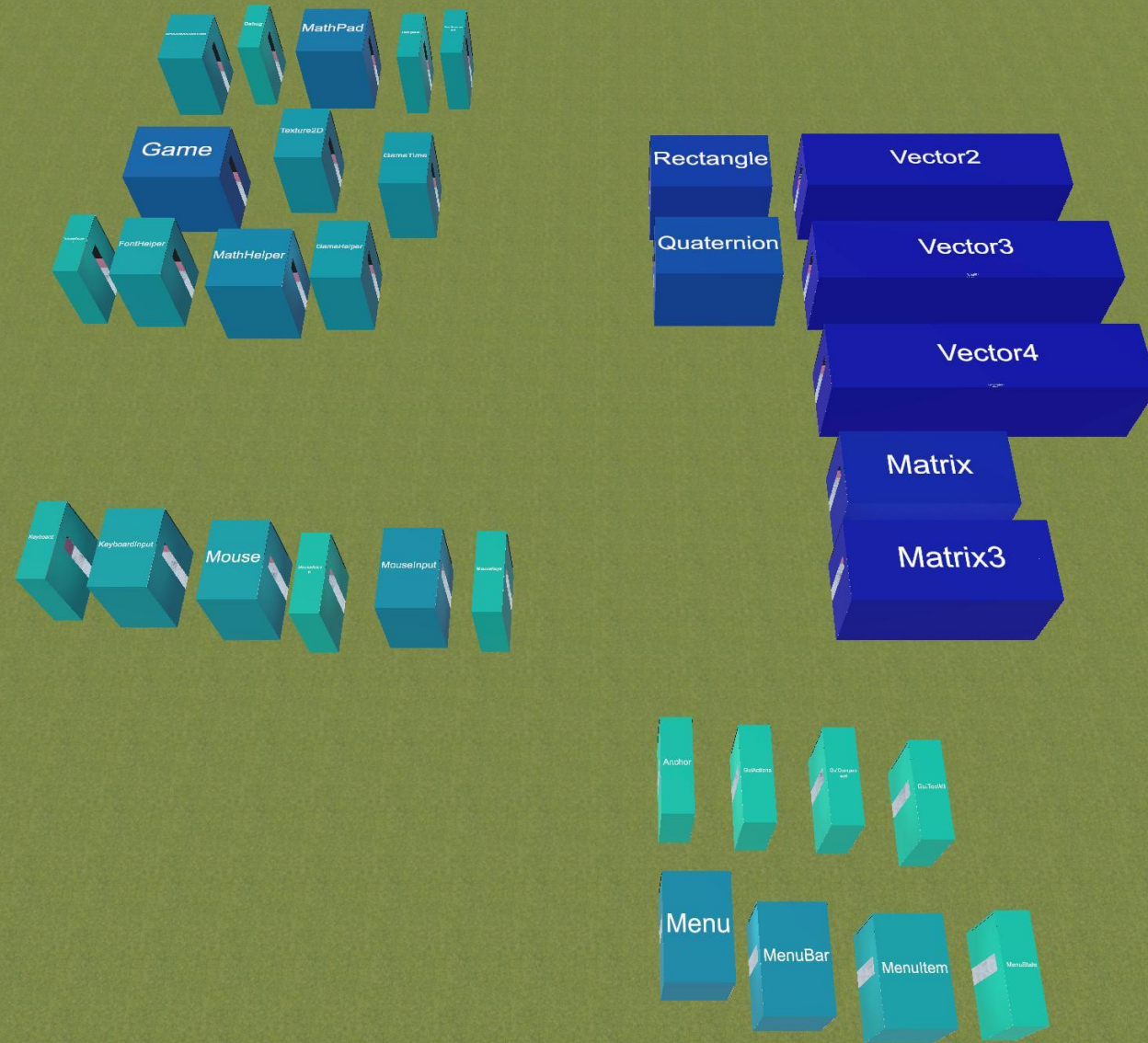
Given project that are already organized into directories



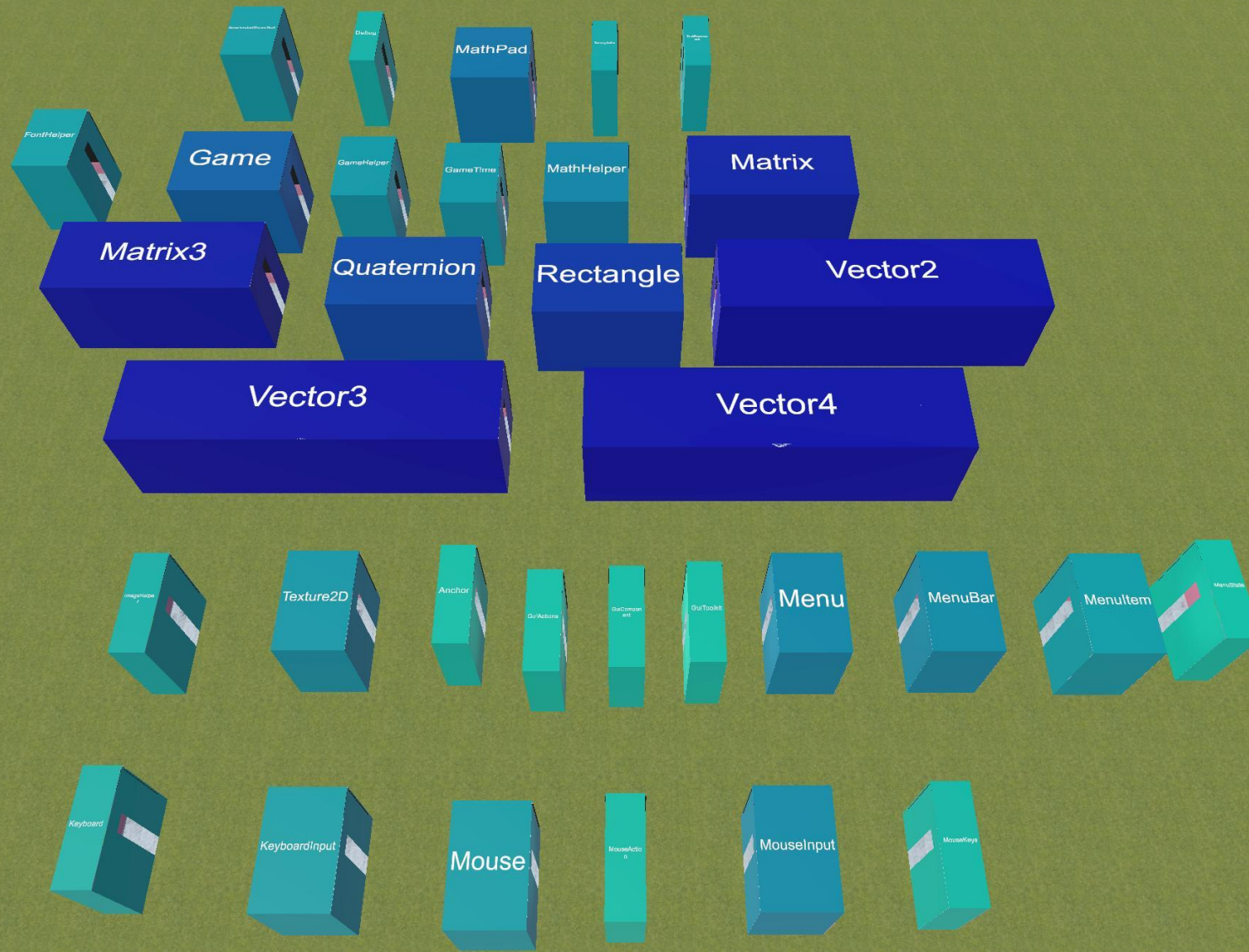


Participant 1: “Folders were arranged spatially in groups. Classes that appeared related by name were sub-grouped.”

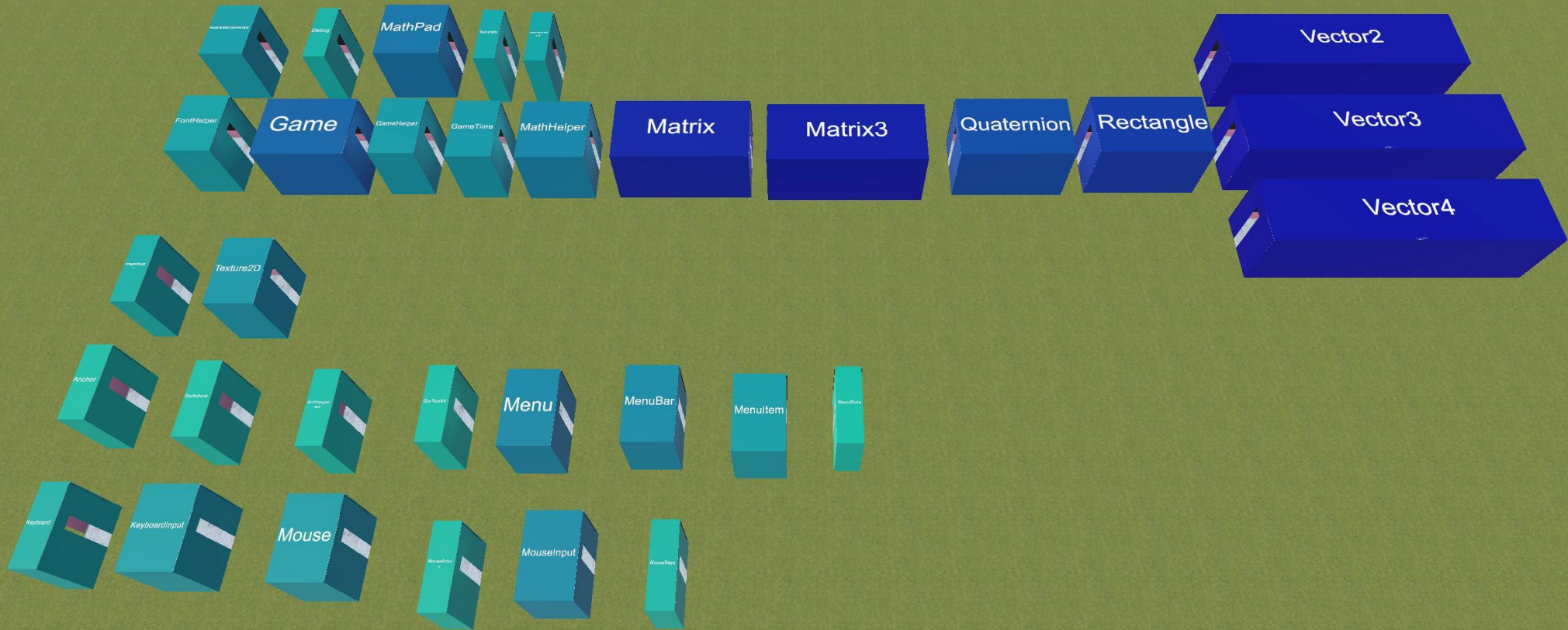






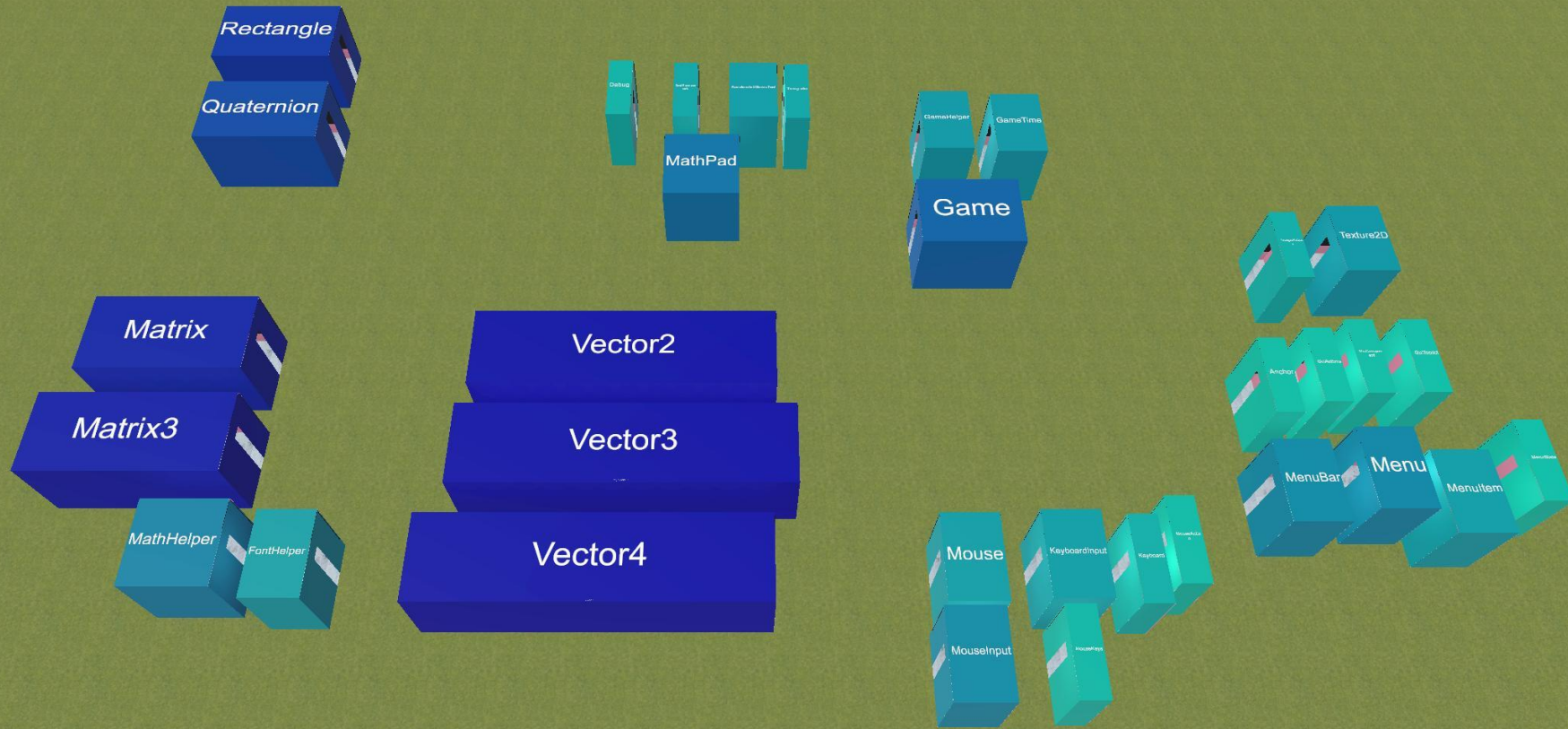






Participant 4: "The classes were arranged alphabetically for each folder and I arranged the classes in the same folder in the same line."





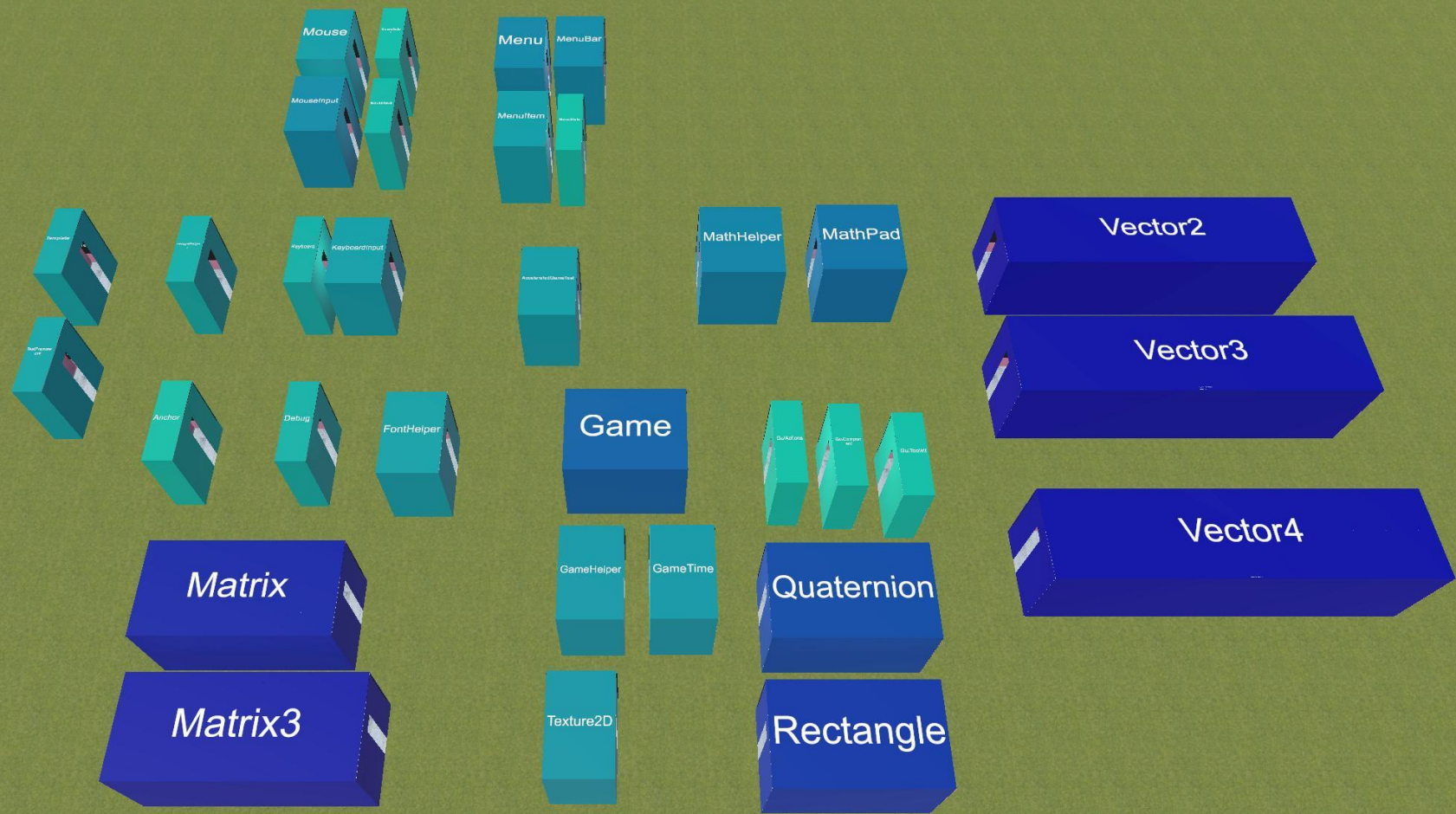
Participant 5: "I tried to group the related class together based on the usefulness and field."



## Group 2

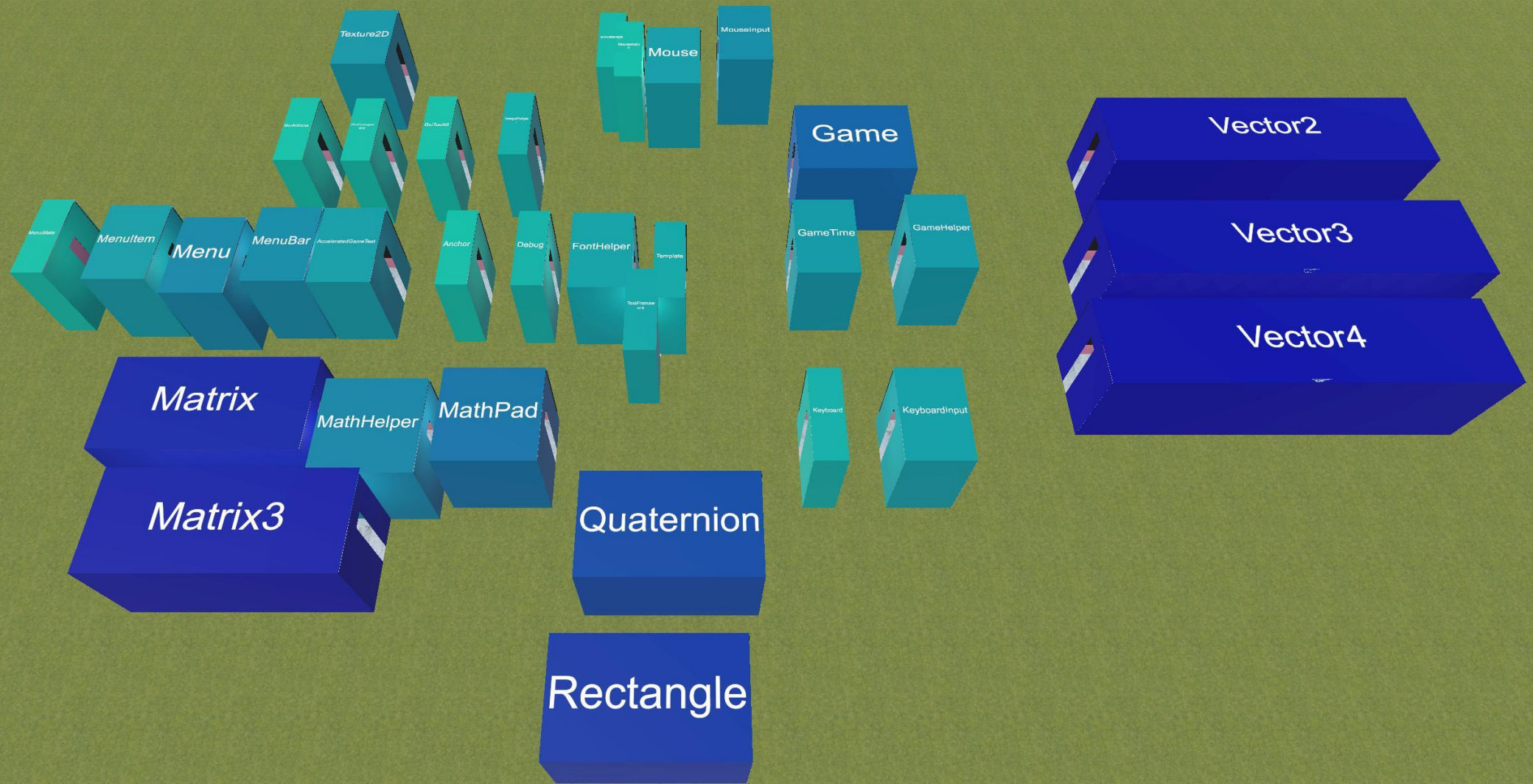
Given project that are not organized in any particular manner





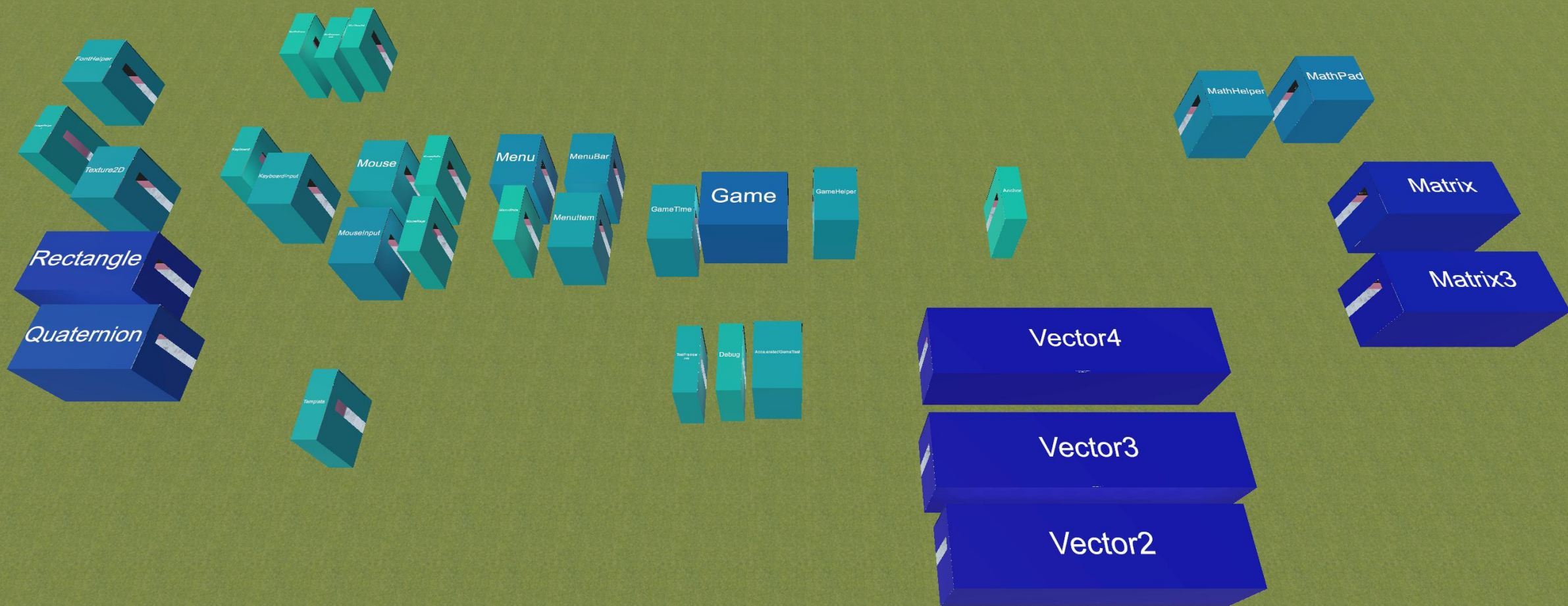
Participant 6: "When arranging the classes my first concern was to group similar classes together."





Participant 7: "I tried to place the rooms in the chunk of similar classes. My priority was to place them in such a way that they are easy to find again."









Participant 9: "Big models together. Smaller ones in the middle so I can find them easier."



## Discussion

- There is a possible relation between the user's cognitive understanding of the codebase and their decisions in organizing building block
- The users mostly chose to organize the constituent parts of the project based on their relationship with respect to each other







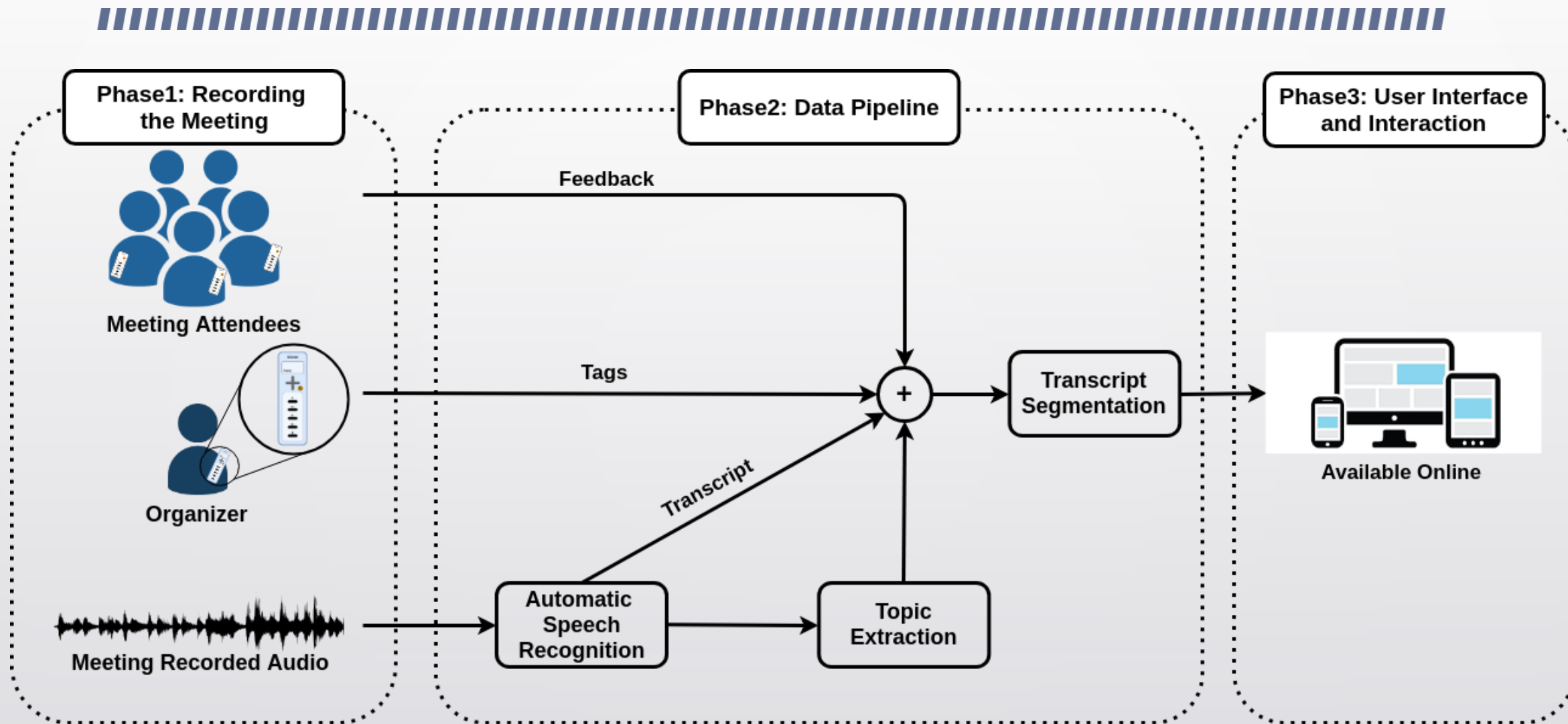
CommunityClick





# Problem

- Notetaking is **distracting** during a meeting
- Collecting all the ideas in a meeting is **hard**







(a)



(b)



(c)



# Demo



# User Study

Study	Objective	Method	Participants
Needfinding	Understating domain problem	Field observation and Survey	20 organizers 66 community members
Study 1 - Controlled Lab Experiment			
Phase 1: Meeting Simulation	Simulating meeting and utilizing iClicker	Observation and Survey	8 participants
Phase 2: Report Creation	Comparing CC with G-Doc	Within-subject comparison	20 participants
Phase 3: Report Evaluation	Evaluating reports	Blind review	5 participants
Study 2 - Expert Feedback	Collecting overall feedback and adoption possibility	Demo and Phone interview	6 civic leaders
In-Class Deployment	Exploring real-world usage	Field observation and Survey	18 students



## Other cool Projects





# Math Boxes 2015





# Electrick 2017



Carnegie  
Mellon  
University

[www.figlab.com](http://www.figlab.com)





# LumiWatch 2018

## LumiWatch

On-Arm Projected Graphics and Touch Input

Robert Xiao

Teng Cao

Ning Guo

Jun Zhuo

Yang Zhang

Chris Harrison

Carnegie Mellon University

ASU



Thank you