ual: creative computing institute

C++ Day 3

Introduce Object-Oriented Programming

Date

Jasper Zheng

Day 3 Activities

9:00 – 9:45 Watch two videos on Object-Oriented Programming

- Introduction to OOP
- Practical tutorial on OOP (3:13:27 to 3:41:42 [Chapter Classes & Objects, Constructor Functions, and Object Functions])

9:45 – Updates on Tic tac toe

Follow the next few slides, we will remake and update the Tic tac toe game

15:30 – 17:00 Q&As [optional]

Day 3 - Activities

Recap: Tic-Tac-Toe (X's and O's Game)

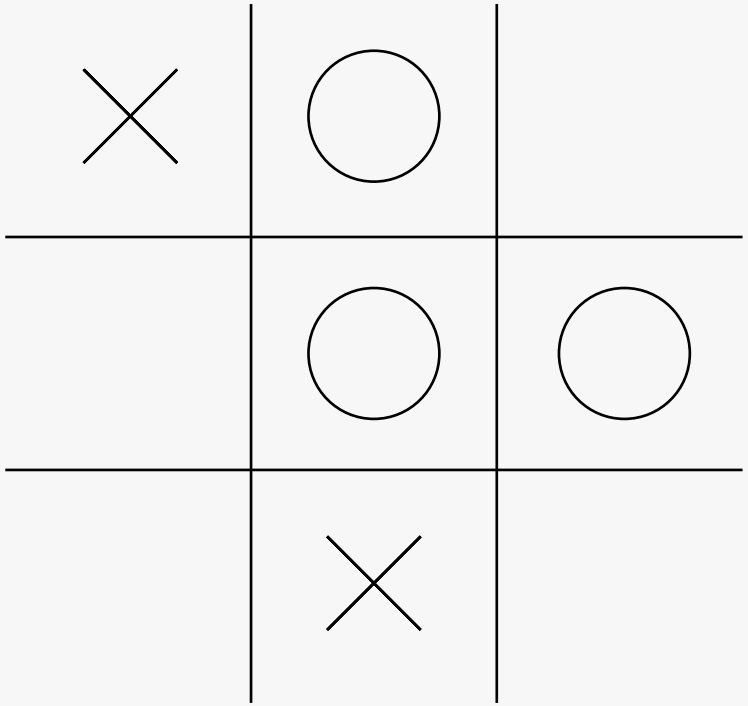
We are making a tic-tac-toe game in the console.

First print a grid to the console.

```
[ ][ ][ ]
[ ][ ][ ]
[ ][ ][ ]
```

Two players take turn entering row and column number to place their marker. After each marker is placed, print out the grid again.

```
Player x
enter row: 1
enter column: 2
[ ] x [ ]
[ ][ ][ ]
[ ][ ][ ]
```



Day 3 - Activities

Recap: Tic-Tac-Toe (X's and O's Game)

Hints

Eventually, the console may look something like this ->

We may have a 2d array/vector representing the grid, (e.g. 0 for empty, 1 for 'x', 2 for 'o'). Player's action will modify data in the array. Then we may use a function to print out the grid according to this array.

Our program will continuously take inputs from the player, so combine the code we've produced in day 1.

How do we decide whose turn is it? The modulo operator in task 2 (it's in day 2) may help.

Do we need to handle exceptions? What if the user enter an invalid row or column number, what if the gird is already taken by the other player?

How to check if a player wins?

```
[ ][ ][ ]
[ ][ ][ ]
[ ][ ][ ]

Player x
enter row: 1
enter column: 2
[ ] x [ ]
[ ][ ][ ]
[ ][ ][ ]

Player o
enter row: 2
enter column: 2
[ ] x [ ]
[ ] o [ ]
[ ][ ][ ]
```

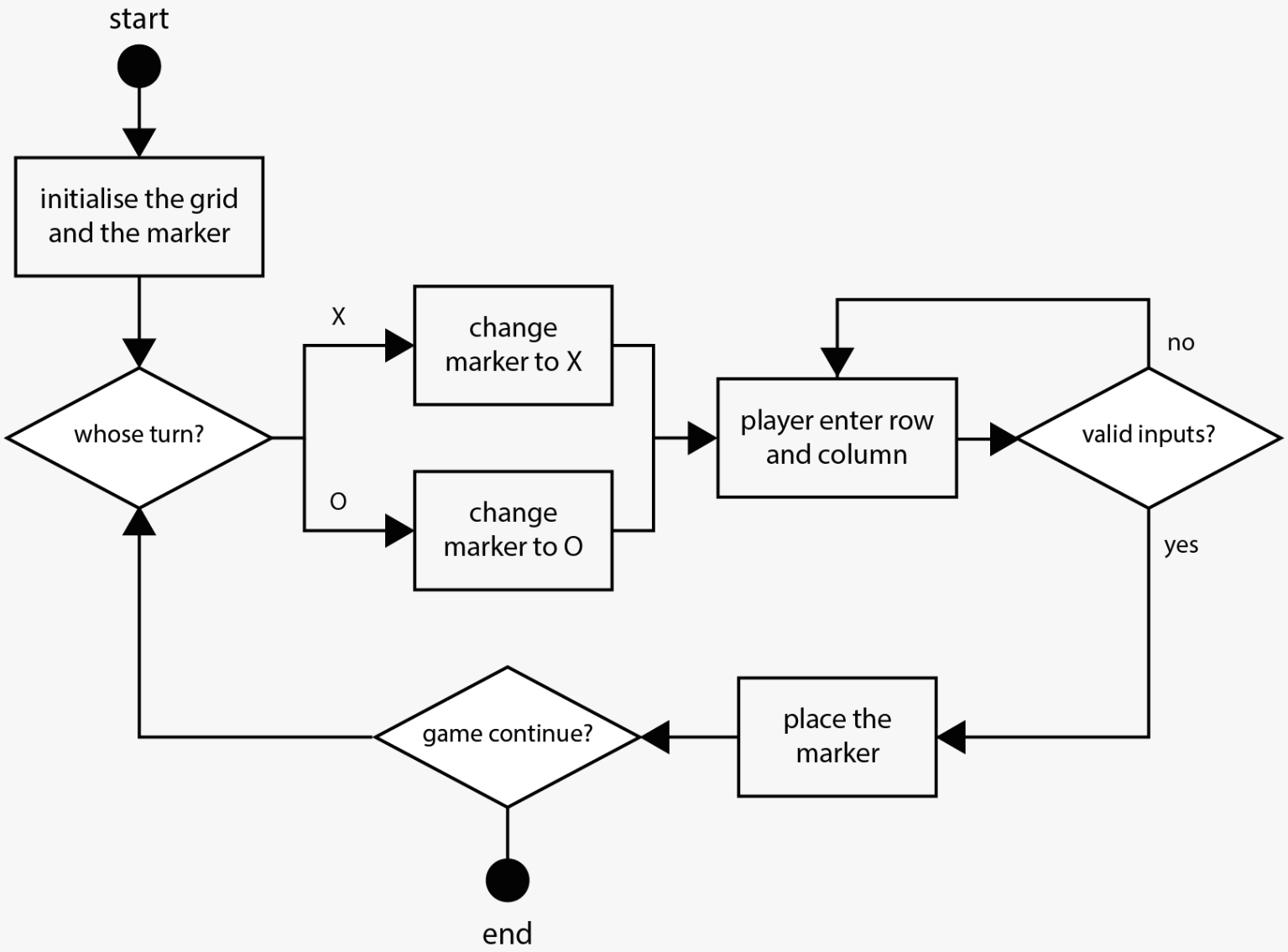
```
Player x
enter row: 1
enter column: 1
 x  x [ ]
[ ] o [ ]
[ ][ ][ ]

Player o
enter row: 1
enter column: 3
 x  x  o
[ ] o [ ]
[ ][ ][ ]

Player x
enter row: 3
enter column: 1
 x  x  o
[ ] o [ ]
 x [ ][ ]
```

Day 3 - Activities

Recap: Flowchart



Day 3 - Activities

System Components

We'll split the system in a Grid object and a Player object.

The grid object handles everything happens on the grid, acting like a referee (i.e. add markers, initialise and print the grid, decide a winner, check if a cell is taken)

The player object represent a player, maintain a player's index and markers, and handle inputs.

```
class Grid;

class Player;
```

class Grid;

Attributes:

```
int x_dim;
int y_dim;
vector<vector<int> >grid;
```

the integer x_dim and y_dim representing the dimension of the grid, in a 3 x 3 grid they'll all be 3

Methods:

the class constructor that is going to run when we define a new Grid object

```
Grid(int x_dim, int y_dim);
void initialiseGrid(int x_dim, int y_dim);
void showGrid();
bool isGameOver();
bool checkInput(int x, int y);
void placeMarker(int x, int y, int marker);

int checkRowCrossed();
int checkColumnCrossed();
int checkDiagonalCrossed();
```

checking on every rows, columns and diagonals to see if there is a winner, possible be called from the `isGameOver()` function (since the game ends when a winner shows up

class Player;

Attributes:

```
int index;
char marker_char;
```

character “X” or “O”

Methods:

```
Interaction(int marker_in);
void playersMove(Grid &grid);
```

we are having a “&” in front of the variable, means we are passing a reference of the vector, instead of passing a copy of it, so changes made in this function reflect in main()

Day 3 - Activities

Take everything into codes

Link to the codes

After we sort everything into classes, we really just want to focus on controlling the flow in our main function, without worry much about internal data and process (e.g. row and column numbers, shifting markers...), so that make out program more maintainable.

```
int main() {

    Grid grid(3, 3);
    grid.showGrid();

    Player player1(1);
    Player player2(2);

    int round = 0;
    while(!grid.isGameOver()){
        if (round%2 == 0){
            player1.playersMove(grid);
        } else{
            player2.playersMove(grid);
        }
        grid.showGrid();
        round+=1;
    }

}
```

Day 3 resources

CreativeApplications.Net: a community of art, media and technology

Quickstart on Github:

Missing Semester: a short course help you to gets familiar with terms like terminal, linux, command-line, git, version controls...

Map of Computer Science: a 10mins video explaining CS

Outlook: C++ at CCI

openFrameworks: a tool for creative coding (e.g. interactive moving images, generative arts / sounds, visualisation...)

Raspberry Pi: run openFrameworks in embedded systems for installations, synthesisers

JUCE: a tool for making music app / plug-ins / virtual instruments

Unity: game engine

Unreal Engine: game engine

Day 3 de-brief

- How was today for you?
- What has gone well?
- What went as planned?
- What surprised you?
- Did you find today difficult?

- Share anything you made?
- Ask around the class and see if they have anything to share?

ual: creative computing institute

Concluding Week 3 Survey

https://artslondon.padlet.org/hbrueggemann/j2yr3zfwkap4v4rq

The password is **Jumpstart**.

Thank you, see you at CCI ☺

ual: creative computing institute

Thank you for joining ☺
Catch you at Welcome Week!

@ual_cci

@ual_cci

arts.ac.uk/cci