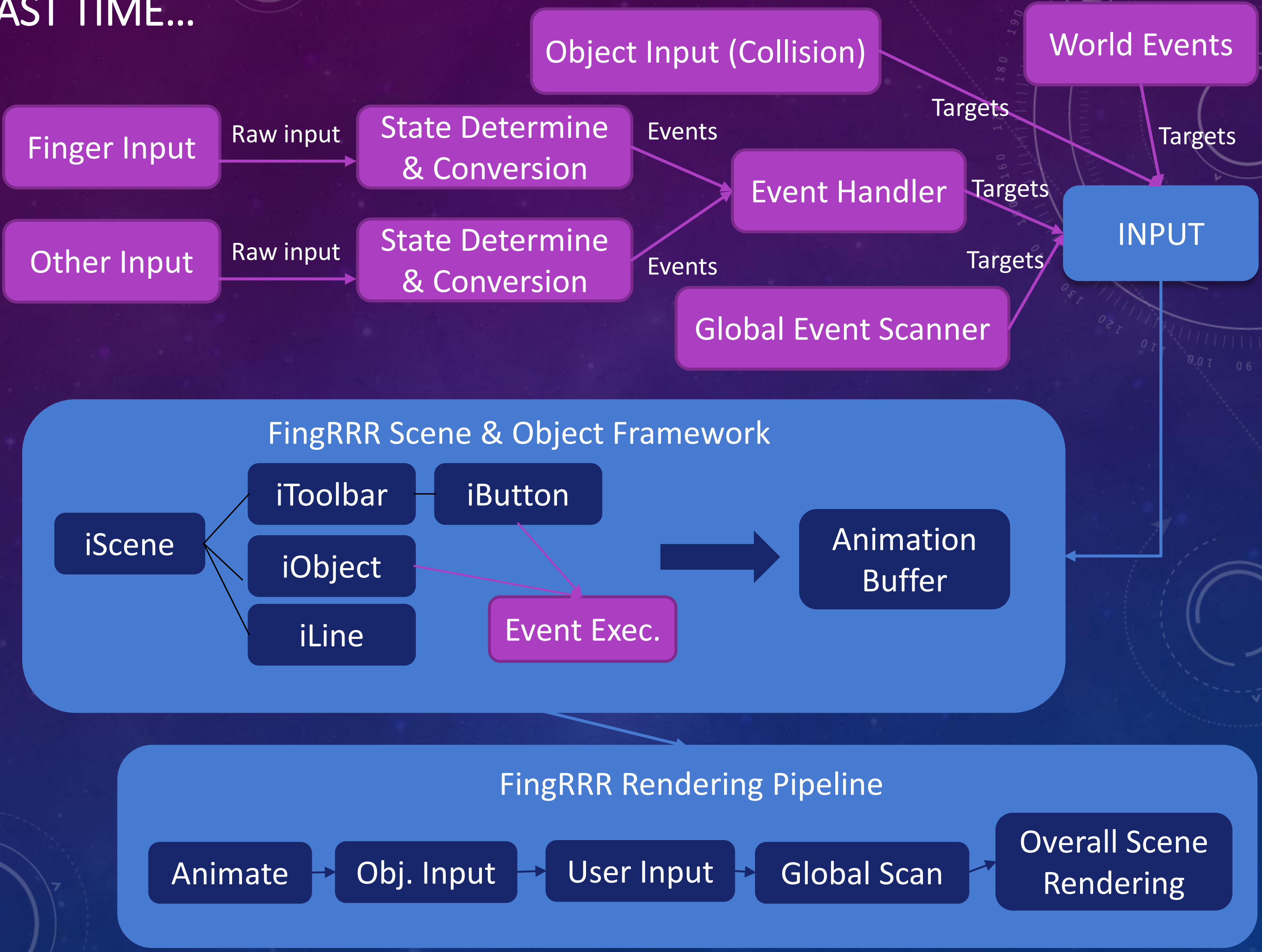
TWITCH PLAYS CONNECT6

A MMO GAME FEATURING TWITCH INTEGRATION

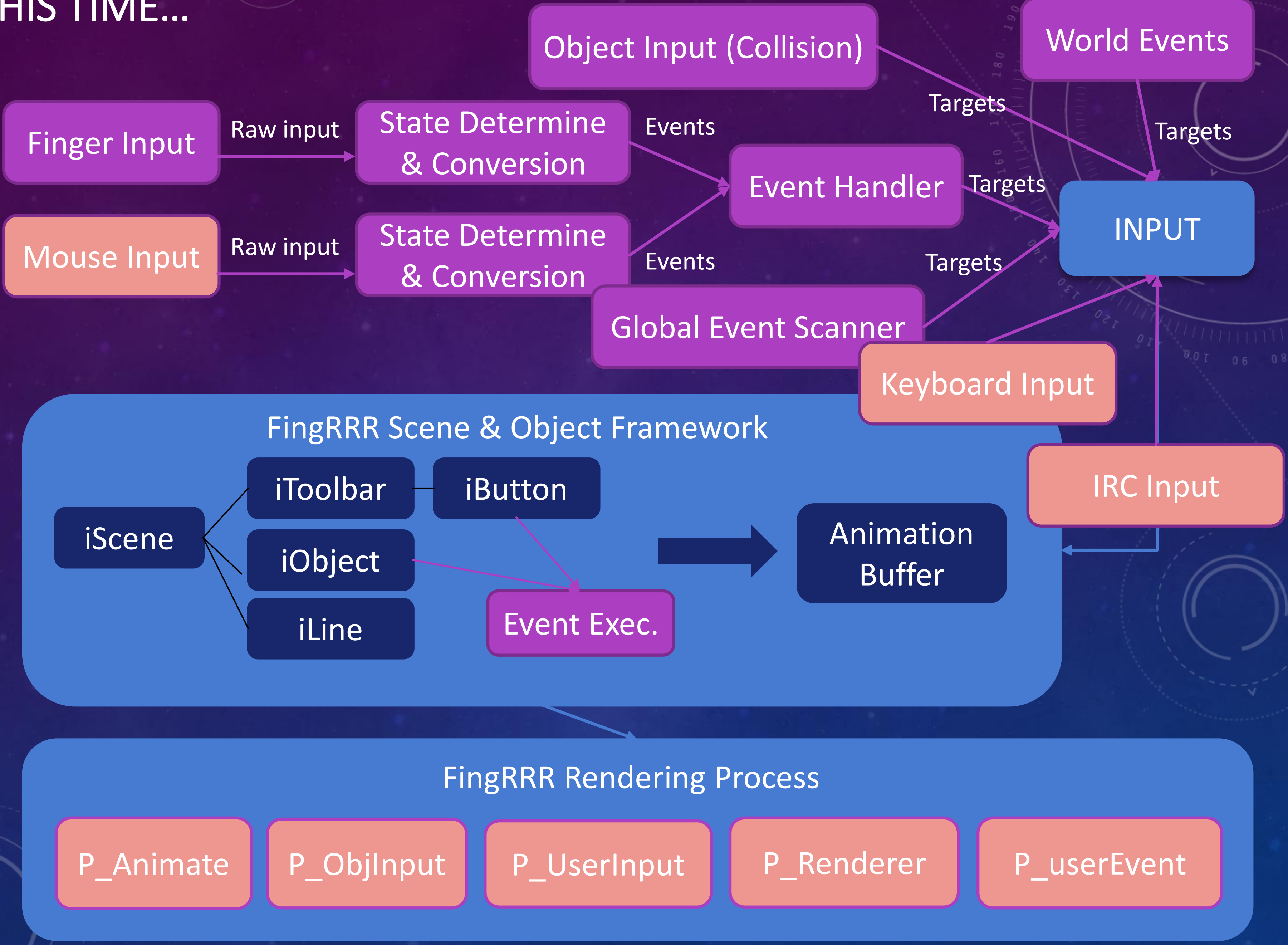
CORE CONCEPTS

- Allow multiple players to participate in a Connect 6 game at the same time.
- Players would receive real-time game window feedback, allowing them to make decisions accordingly. The decisions, or commands are issued via chat server (IRC), which is one of the unique part.

LAST TIME…


Object Input (Collision)
World Events
Targets
Targets
Finger Input
Raw input
State Determine & Conversion
Events
Event Handler
Targets
INPUT
Other Input
Raw input
State Determine & Conversion
Events
Targets
Global Event Scanner
FingRRR Scene & Object Framework
iScene
iToolbar
iButton
iObject
iLine
Event Exec.
Animation Buffer
FingRRR Rendering Pipeline
Animate
Obj. Input
User Input
Global Scan
Overall Scene Rendering

THIS TIME…


Object Input (Collision)
World Events
Finger Input
Raw input
State Determine & Conversion
Events
Targets
Targets
Event Handler
Targets
INPUT
Mouse Input
Raw input
State Determine & Conversion
Events
Targets
Global Event Scanner
Keyboard Input
FingRRR Scene & Object Framework
iToolbar
iButton
iScene
iObject
iLine
Event Exec.
Animation Buffer
IRC Input
FingRRR Rendering Process
P_Animate
P_ObjInput
P_UserInput
P_Renderer
P_userEvent

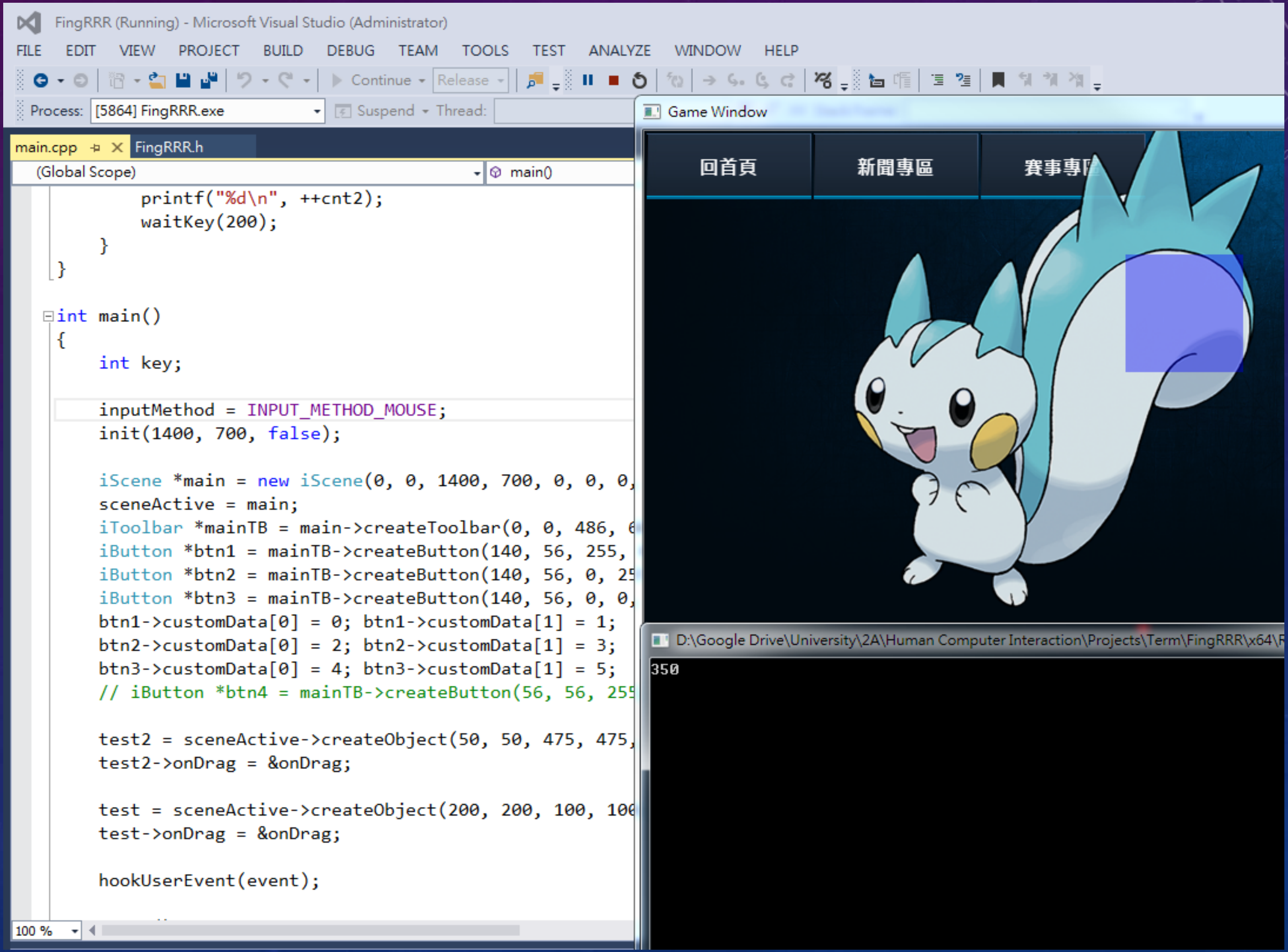
TERMINOLOGY

- Twitch (TV): a popular streaming platform, features IRC chat as their chat service
- Connect6: a famous puzzle game invented by Prof. I-Chen Wu from NCTU. The one who gets six or more stones in a row (horizontally, vertically or diagonally) first wins the game.

REFERENCES

- Twitch Plays Pokémon (2014)
  Twitch Plays Pokémon is a social experiment and channel on the video streaming website Twitch, consisting of a crowdsourced attempt to play Game Freak's and Nintendo's Pokémon video games by parsing commands sent by users through the channel's chat room.
  (http://en.wikipedia.org/wiki/Twitch_Plays_Pok%C3%A9mon)
- FingRRR (2014~2015)
  FingRRR is an extensible 2-D interaction framework structured on OpenCV v2.x written by Inishan (Me). The framework is extensible, thus allowing multiple input and other customization. (2015)
- C++ Console IRC Client (2011~2014)
  C++ Console IRC Client, as its name, is an IRC client written by Fredi Machado.
  (https://github.com/Fredi/IRCClient)

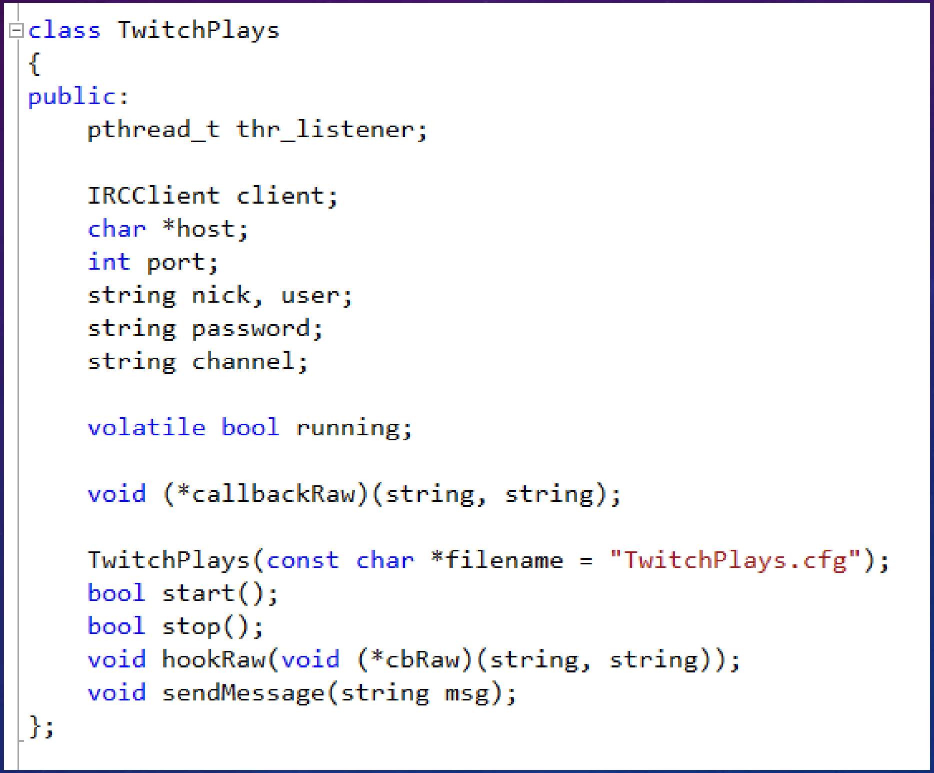
FINGRRR


FingRRR (Running) - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Continue Release
Process: [5864] FingRRR.exe Suspend Thread:
main.cpp FingRRR.h
(Global Scope) main()
Game Window
回首頁
新聞專區
賽事專
D:\Google Drive\University\2A\Human Computer Interaction\Projects\Term\FingRRR\x64\R
100 %

```
        printf("%d\n", ++cnt2);
        waitKey(200);
    }
}

int main()
{
    int key;

    inputMethod = INPUT_METHOD_MOUSE;
    init(1400, 700, false);

    iScene *main = new iScene(0, 0, 1400, 700, 0, 0, 0,
    sceneActive = main;
    iToolbar *mainTB = main->createToolbar(0, 0, 486, 6
    iButton *btn1 = mainTB->createButton(140, 56, 255,
    iButton *btn2 = mainTB->createButton(140, 56, 0, 25
    iButton *btn3 = mainTB->createButton(140, 56, 0, 0,
    btn1->customData[0] = 0; btn1->customData[1] = 1;
    btn2->customData[0] = 2; btn2->customData[1] = 3;
    btn3->customData[0] = 4; btn3->customData[1] = 5;
    // iButton *btn4 = mainTB->createButton(56, 56, 255

    test2 = sceneActive->createObject(50, 50, 475, 475,
    test2->onDrag = &onDrag;

    test = sceneActive->createObject(200, 200, 100, 100
    test->onDrag = &onDrag;

    hookUserEvent(event);
```

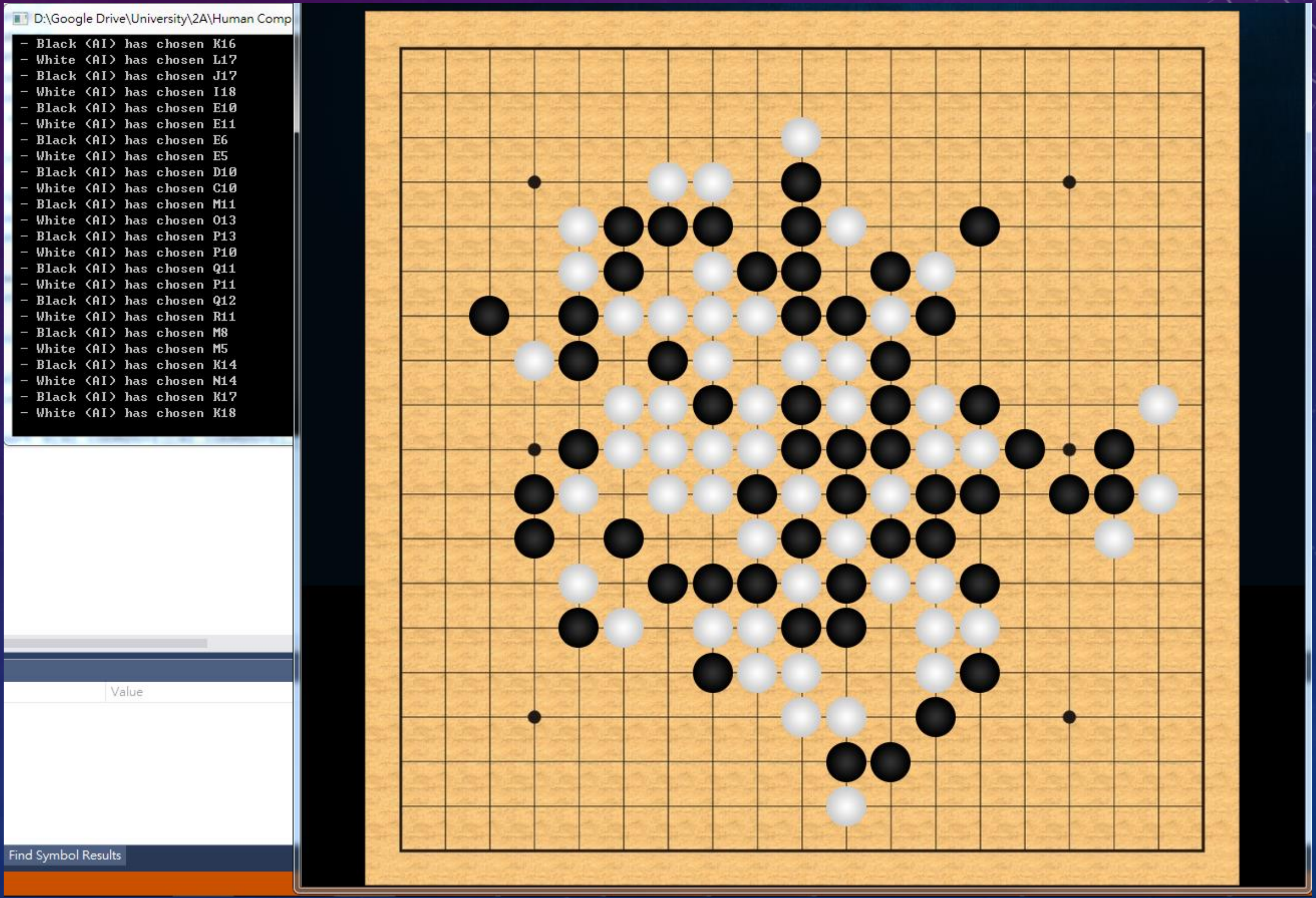
```
350
```

TWITCH PLAYS API



```
class TwitchPlays
{
public:
    pthread_t thr_listener;

    IRCClient client;
    char *host;
    int port;
    string nick, user;
    string password;
    string channel;

    volatile bool running;

    void (*callbackRaw)(string, string);

    TwitchPlays(const char *filename = "TwitchPlays.cfg");
    bool start();
    bool stop();
    void hookRaw(void (*cbRaw)(string, string));
    void sendMessage(string msg);
};
```

ARTIFICIAL INTELLIGENCE



D:\Google Drive\University\2A\Human Comp

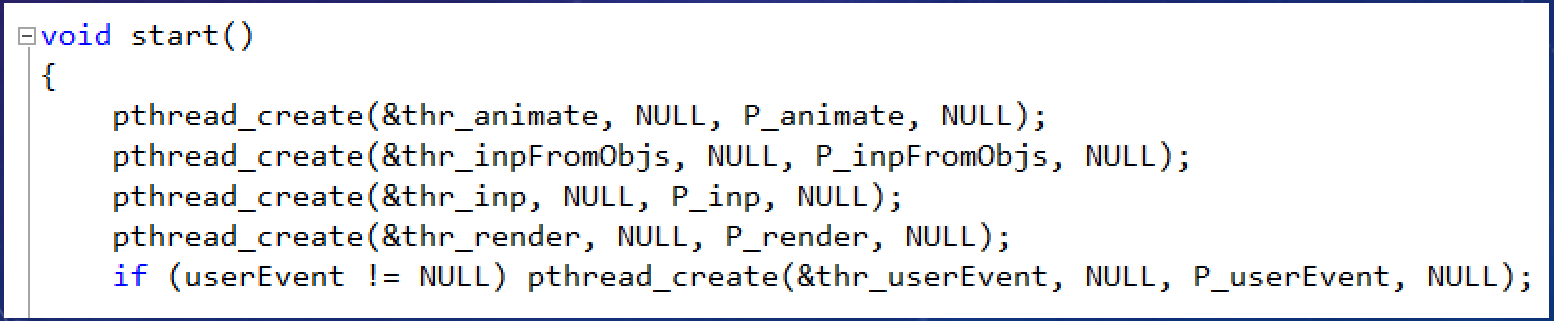
```
- Black (AI) has chosen K16
- White (AI) has chosen L17
- Black (AI) has chosen J17
- White (AI) has chosen I18
- Black (AI) has chosen E10
- White (AI) has chosen E11
- Black (AI) has chosen E6
- White (AI) has chosen E5
- Black (AI) has chosen D10
- White (AI) has chosen C10
- Black (AI) has chosen M11
- White (AI) has chosen O13
- Black (AI) has chosen P13
- White (AI) has chosen P10
- Black (AI) has chosen Q11
- White (AI) has chosen P11
- Black (AI) has chosen Q12
- White (AI) has chosen R11
- Black (AI) has chosen M8
- White (AI) has chosen M5
- Black (AI) has chosen K14
- White (AI) has chosen N14
- Black (AI) has chosen K17
- White (AI) has chosen K18
```

TECHNIQUES

- Multithreaded Programming
- Network Programming
- Artificial Intelligence

MULTITHREADED PROGRAMMING

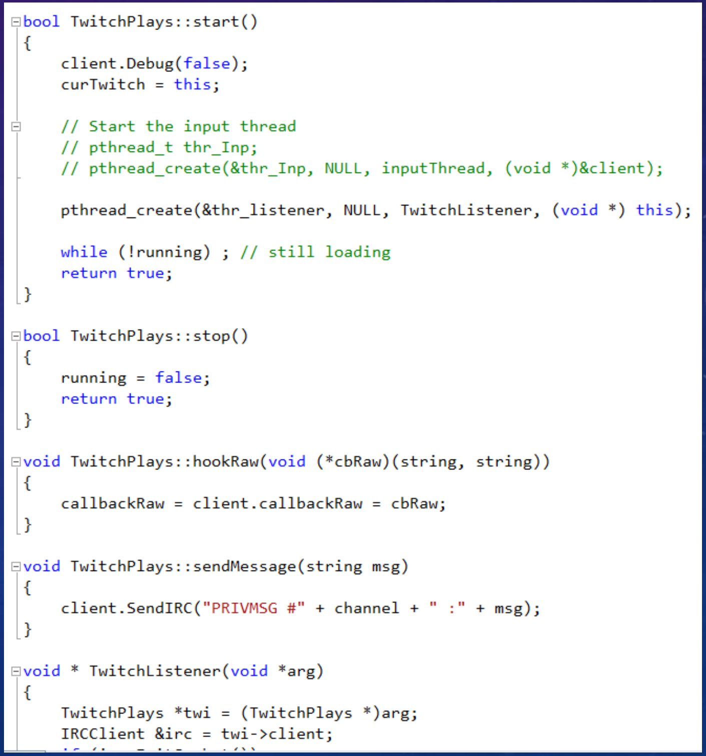
- FingRRR itself uses 6 threads, each addresses main(keyboard input), animation, object interaction, finger and mouse input, renderer and user event, respectively.
- TwitchPlaysAPI uses 2+ threads, 1 serves as a listener to receive IRC messages and others are used for network programming.



```
void start()
{
    pthread_create(&thr_animate, NULL, P_animate, NULL);
    pthread_create(&thr_inpFromObjs, NULL, P_inpFromObjs, NULL);
    pthread_create(&thr_inp, NULL, P_inp, NULL);
    pthread_create(&thr_render, NULL, P_render, NULL);
    if (userEvent != NULL) pthread_create(&thr_userEvent, NULL, P_userEvent, NULL);
```

NETWORK PROGRAMMING

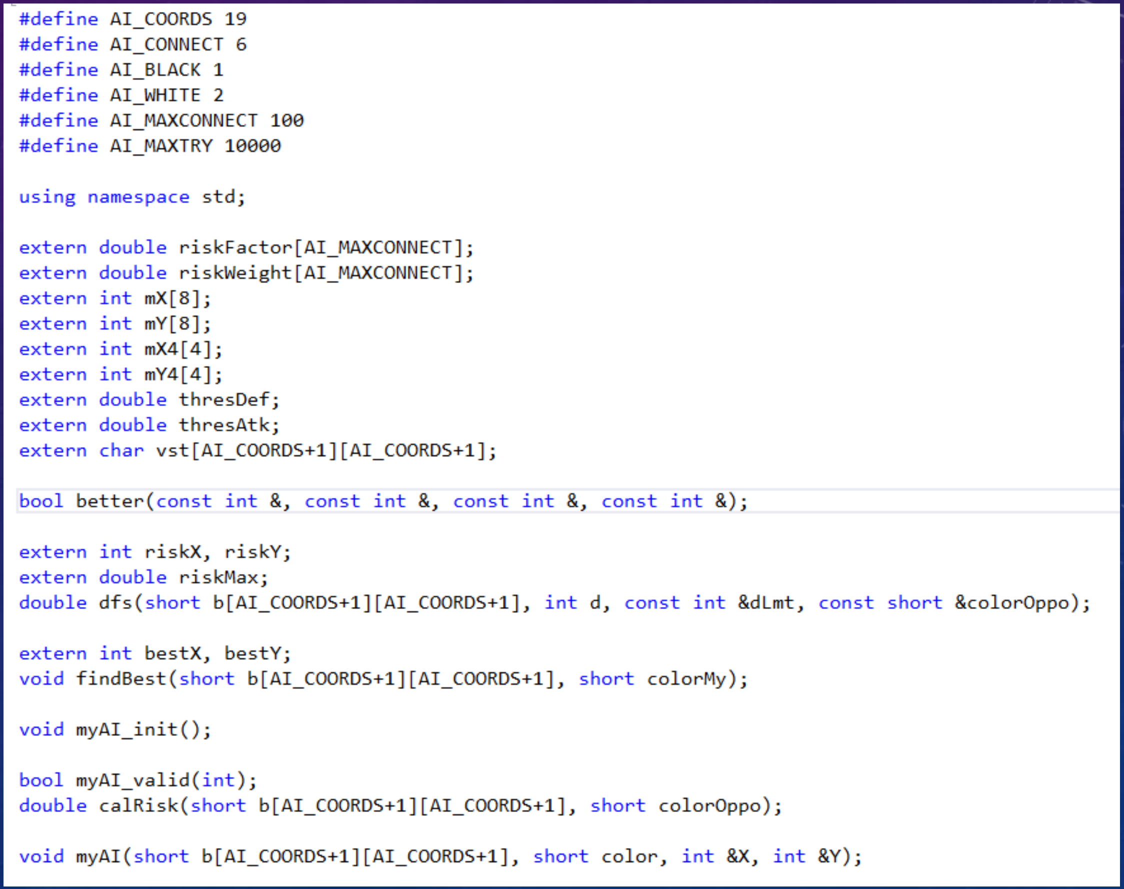
- I didn’t touch too much on this part, but I did modify the IRC Client quite a bit (fixed some bugs, too) to fit my needs.



```
bool TwitchPlays::start()
{
    client.Debug(false);
    curTwitch = this;

    // Start the input thread
    // pthread_t thr_Inp;
    // pthread_create(&thr_Inp, NULL, inputThread, (void *)&client);

    pthread_create(&thr_listener, NULL, TwitchListener, (void *) this);

    while (!running) ; // still loading
    return true;
}

bool TwitchPlays::stop()
{
    running = false;
    return true;
}

void TwitchPlays::hookRaw(void (*cbRaw)(string, string))
{
    callbackRaw = client.callbackRaw = cbRaw;
}

void TwitchPlays::sendMessage(string msg)
{
    client.SendIRC("PRIVMSG #" + channel + " :" + msg);
}

void * TwitchListener(void *arg)
{
    TwitchPlays *twi = (TwitchPlays *)arg;
    IRCClient &irc = twi->client;
```

ARTIFICIAL INTELLIGENCE

- Algorithms including Iterative Deepening and Heuristic Search.



```
#define AI_COORDS 19
#define AI_CONNECT 6
#define AI_BLACK 1
#define AI_WHITE 2
#define AI_MAXCONNECT 100
#define AI_MAXTRY 10000

using namespace std;

extern double riskFactor[AI_MAXCONNECT];
extern double riskWeight[AI_MAXCONNECT];
extern int mX[8];
extern int mY[8];
extern int mX4[4];
extern int mY4[4];
extern double thresDef;
extern double thresAtk;
extern char vst[AI_COORDS+1][AI_COORDS+1];

bool better(const int &, const int &, const int &, const int &);

extern int riskX, riskY;
extern double riskMax;
double dfs(short b[AI_COORDS+1][AI_COORDS+1], int d, const int &dLmt, const short &colorOppo);

extern int bestX, bestY;
void findBest(short b[AI_COORDS+1][AI_COORDS+1], short colorMy);

void myAI_init();

bool myAI_valid(int);
double calRisk(short b[AI_COORDS+1][AI_COORDS+1], short colorOppo);

void myAI(short b[AI_COORDS+1][AI_COORDS+1], short color, int &X, int &Y);
```

ADVANTAGES / UNIQUENESS

- Allows massive players to participate in a single game simultaneously.
- Highly cooperative and fun
- The interaction method itself is pretty unique
- The interaction method can potentially be useful in other areas. For instance, education and enterprises.
- I wrote most of the parts on my own !
  FingRRR: ~1500 lines.
  TwitchPlaysAPI: ~300 lines
  Connect6 AI: ~300 lines.

CONTRIBUTION

- 0113110 陳柏翰 – handles all parts of this term project.

Thanks for listening !