Int egrado;

Robocup CoSpace

Demonstration

EQUIPE

```
void Integrantes()
{
    Felipe Batista Martins;
    Kauê Cano Souza;
    Luis Fernando Wadt Assis;
    Matheus Henrique de
    Almeida Camacho;
}
```

Colégio Integrado Objetivo
São Paulo – SP
Brasil

O DESAFIO

- Localizar objetos;
- Área laranja (“caixa de coleta”);
- Paredes, obstáculos e armadilhas;
- Dois mapas;

PROBLEMAS GERAIS

- Falta de painel de debug;
- Posição dos sensores de ultrassom;
- Informações escassas para fazer um mapeamento decente;
- Plataforma limitante;
- Linguagem de programação C#;

IDEIAS INICIAIS

Códigos propostos:

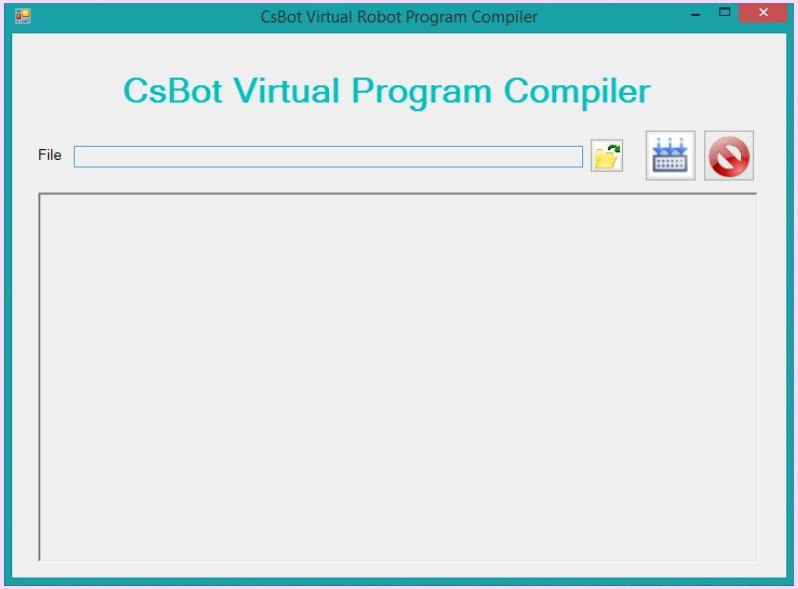
- “Pelancudo”

Código baseado em IA, o qual “aprende” a cada execução dentro da arena proposta

- “Renatão”

O código propõe um mapeamento da arena ao se iniciar a partida

COMO SEGUIR A PROPOSTA?



- Escrever o código inicialmente em C++;
- Aprender C# (única linguagem suportada pela plataforma);
- Transcrever o código de C++ para C#;

Vantagens do uso de código em linha
Maior liberdade na montagem de uma estratégia
Facilidade na execução de cálculos matemáticos mais complexos
Criação mais rápida e simples de um código mais requintado

PELANCUDO

Código baseado em IA (neurônios)	
Prós	Eficiência indiscutível
	A cada geração, aprende melhor a arena
	Média da pontuação obtida otimizada a cada geração

```
#ifndef NEURONIO
#define NEURONIO

#include <vector>

#define MAX_QT_POWER 25
#define K_STR 10

class Neuron;
enum tipo_neuronio  {
    SENSITIVO, MOTOR, CONECTOR
};

struct connection   {
    double k_strength[K_STR];
    Neuron * c_neuronio;
};

struct connection_to_make   {
    double k_strength[K_STR];
    char identifier[16];
};
```

Início do código “Pelancudo”

PELANCUDO

Código baseado em IA (neurônios)	
Contras	Bastante tempo e processamento para analisar a arena
	Exige várias execuções antes da rodada oficial na arena a ser usada
	Pequenos erros humanos têm grandes efeitos

```
class Neuron    {
    private:
        double charge;
        double vcharge[MAX_QT_POWER];
        float threshold;
        char name[256];
    public:
        std::vector<connection> conexoes;
        char identifier[16];
        tipo_neuronio tipo;

        // se tipo == SENSITIVo, charge_id deve ser diferente de 0.
        // todos os neuronios motores tem action_id.
        int charge_id;
        int action_id;

        void set_charge();

        Neuron();
        Neuron(char * a, tipo_neuronio tipo, float threshold);
        Neuron(const char*);
        ~Neuron();

        void output_to_file(const char*);
        void output_to_file(const char*, const char*);
        std::vector<connection_to_make> load_from_file(const char * arquivo);

        void make_connection(Neuron&, const double *);
};

void make_connection(Neuron&, Neuron&);
#endif /* NEURONIO */
```

Parte do código “Pelancudo”

RENATÃO

Mapeamento da arena	
Prós	Analisa qualquer tipo de arena em segundos
	Evita que o robô se perca na arena e, assim, choque-se contra uma parede
	Após analisada a área, o movimento do robô é bem mais seguro

```
const byte SUPER = 9;
const byte BASE = 10;

public static int AnguloDestino = Compass;
private static double pX = 0, pY = 0;
static byte[,] mapa = new byte[4*TMX,4*TMY];

public static void virar(int angulo, int R, int L) {
    AnguloDestino = (Compass + angulo)%360;
    Wheel_Left = L;
    Wheel_Right = R;
}
public static bool achouObjeto()    {
    return false;
}
public static void collect()    {

}

public static void OnTimer()
{
    if(!firstTime)
        Game0();
    else    {
        for(int i=0;i<4*TMX;++i)
            for(int j=0;j<4*TMY;++j)
                mapa[i,j] = 0;
```

Parte do código “Renatão”

RENATÃO

Mapeamento da arena	
Contras	Ainda assim, é limitado
	Exige um tempo inicial para análise do local desconhecido
	Não analisa todas as áreas numa primeira tentativa, o que exige mais de uma análise por arena

```
const byte SUPER = 9;
const byte BASE = 10;

public static int AnguloDestino = Compass;
private static double pX = 0, pY = 0;
static byte[,] mapa = new byte[4*TMX,4*TMY];

public static void virar(int angulo, int R, int L) {
    AnguloDestino = (Compass + angulo)%360;
    Wheel_Left = L;
    Wheel_Right = R;
}
public static bool achouObjeto()    {
    return false;
}
public static void collect()    {

}

public static void OnTimer()
{
    if(!firstTime)
        Game0();
    else    {
        for(int i=0;i<4*TMX;++i)
            for(int j=0;j<4*TMY;++j)
                mapa[i,j] = 0;
```

Parte do código “Renatão”

O QUE SE APRENDEU

- Aplicação do conhecimento teórico em diversas áreas;

- Experiência que pode ser reaproveitada em futuras competições;

- Potencial e importância do CoSpace.

AGRADECIMENTOS

Renato “r3n4tÃ0” Pacheco

Prof. Caniloi

Malu