

Platform for the realization of interactive group activities in teaching

Matúš Valko

LAB OF SOFTWARE ARCHITECTURES
AND INFORMATION SYSTEMS

FACULTY OF INFORMATICS
MASARYK UNIVERSITY, BRNO



Content



Introduction



Godot



Solution



Problems & needs to be done



Ideas for the future



Introduction

Experience from (boring) lectures



Introduction

Experience from (boring) lectures

Experience from (joyful) lectures



Introduction

- Merging ideas into one solution


Introduction

- Merging ideas into one solution

Code puzzle



Coding competition



Quiz questions

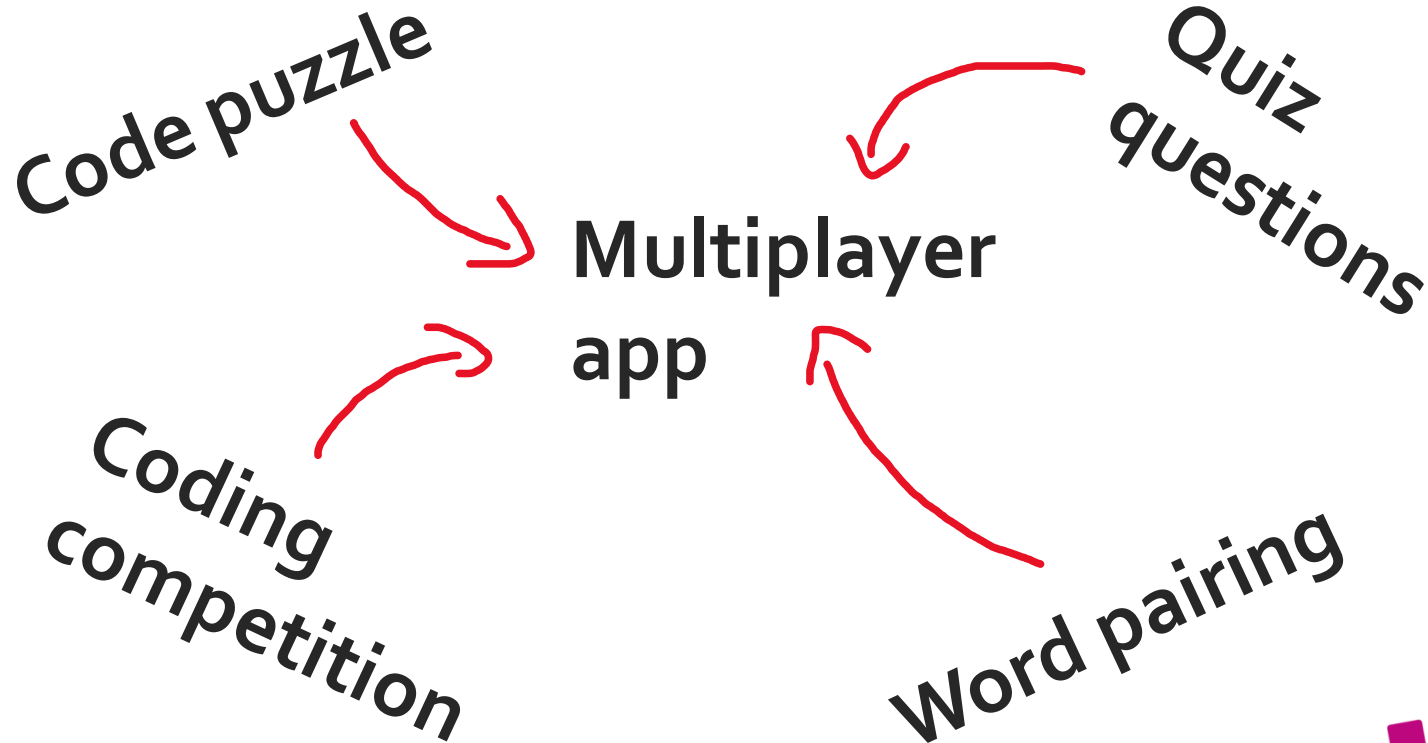


Word pairing



Introduction

- Merging ideas into one solution



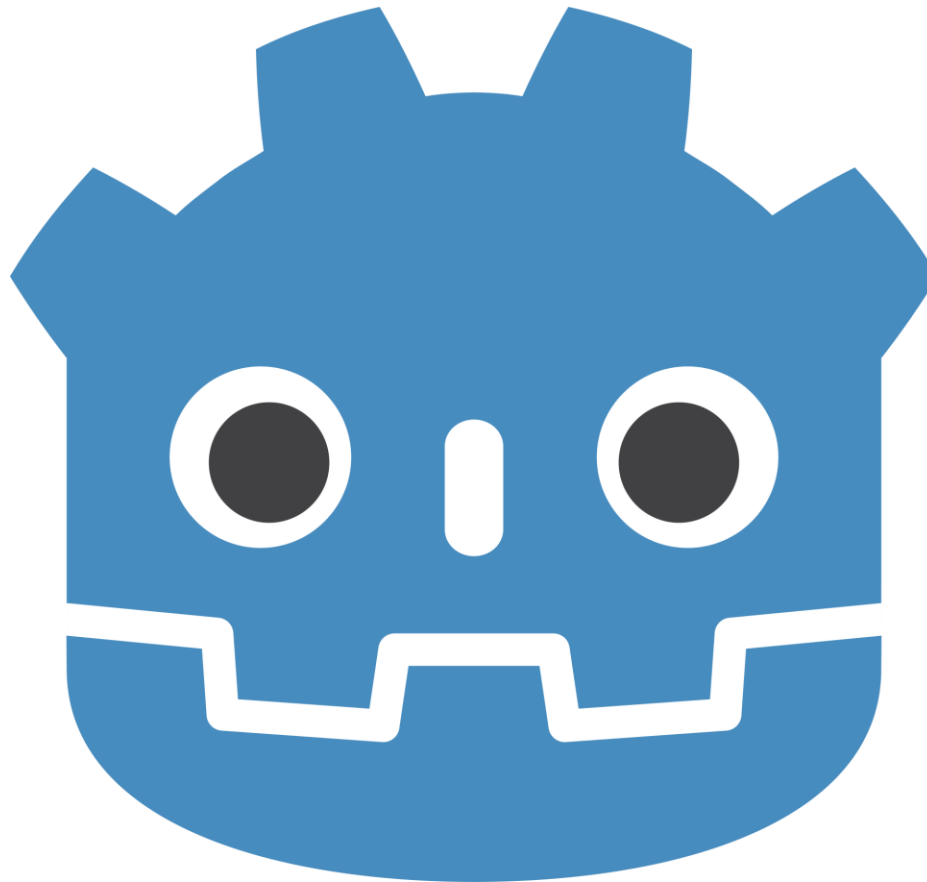
Introduction

- Which framework/engine? (Unity, Blazor, Godot)

Introduction

- Which framework/engine? (Unity, Blazor, Godot)
- Small project, easy deploy, interactive multiplayer

Godot



Godot

Cross – platform
(pc, mobile, web)

Community

Opensource

Tree of nodes

GScript (optim.
For godot scene-
based
architecture), C#

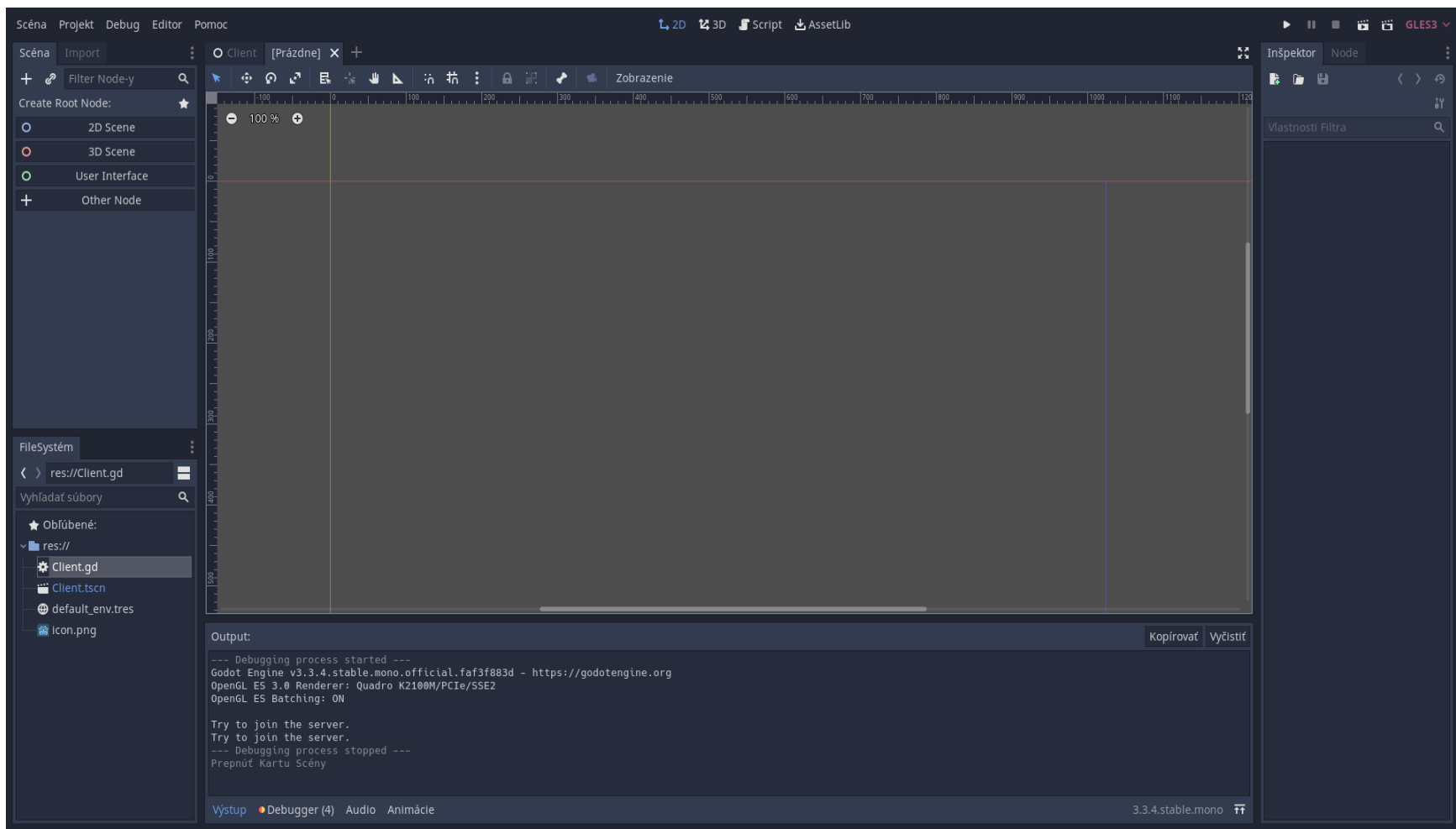
OpenGL
(rendering)

Godot

- Networking
 - (UDP, TCP)
 - (SSL, HTTP)
 - Uses some mid-level

- NetworkedMultiplayerPeer
- Rpc / rpc_id

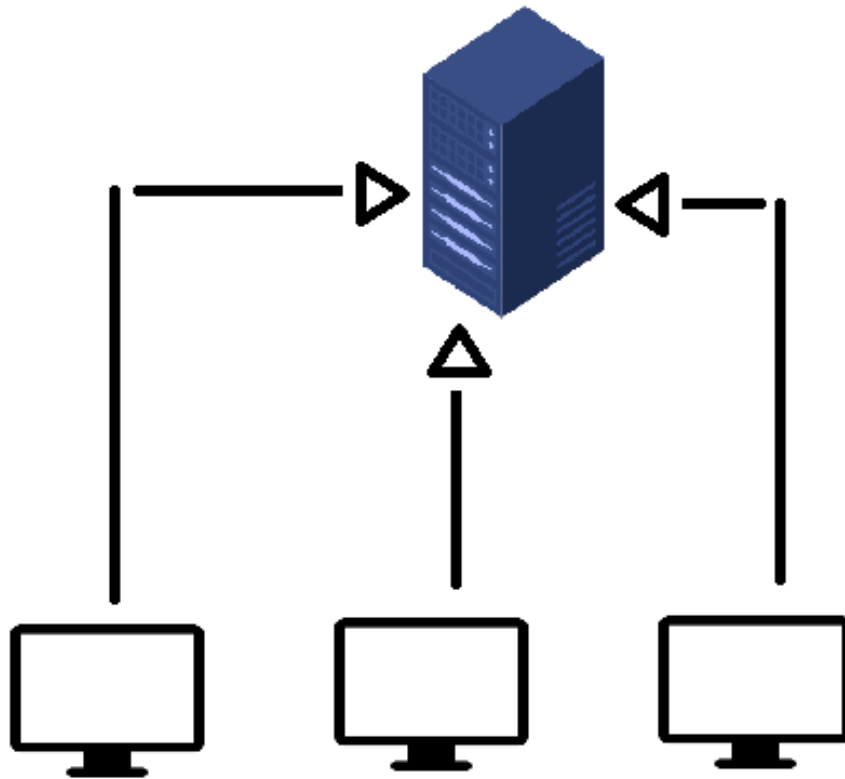
Godot



Solution

- 2 parts – Server / Client
 - Server to be deployed
 - Client as the main part
- Server
 - Awaiting players connecting
 - Singleton (manager) to manage games and players
- Client
 - All the lobby/game logic
 - Singleton (manager) to manage/create games

Solution



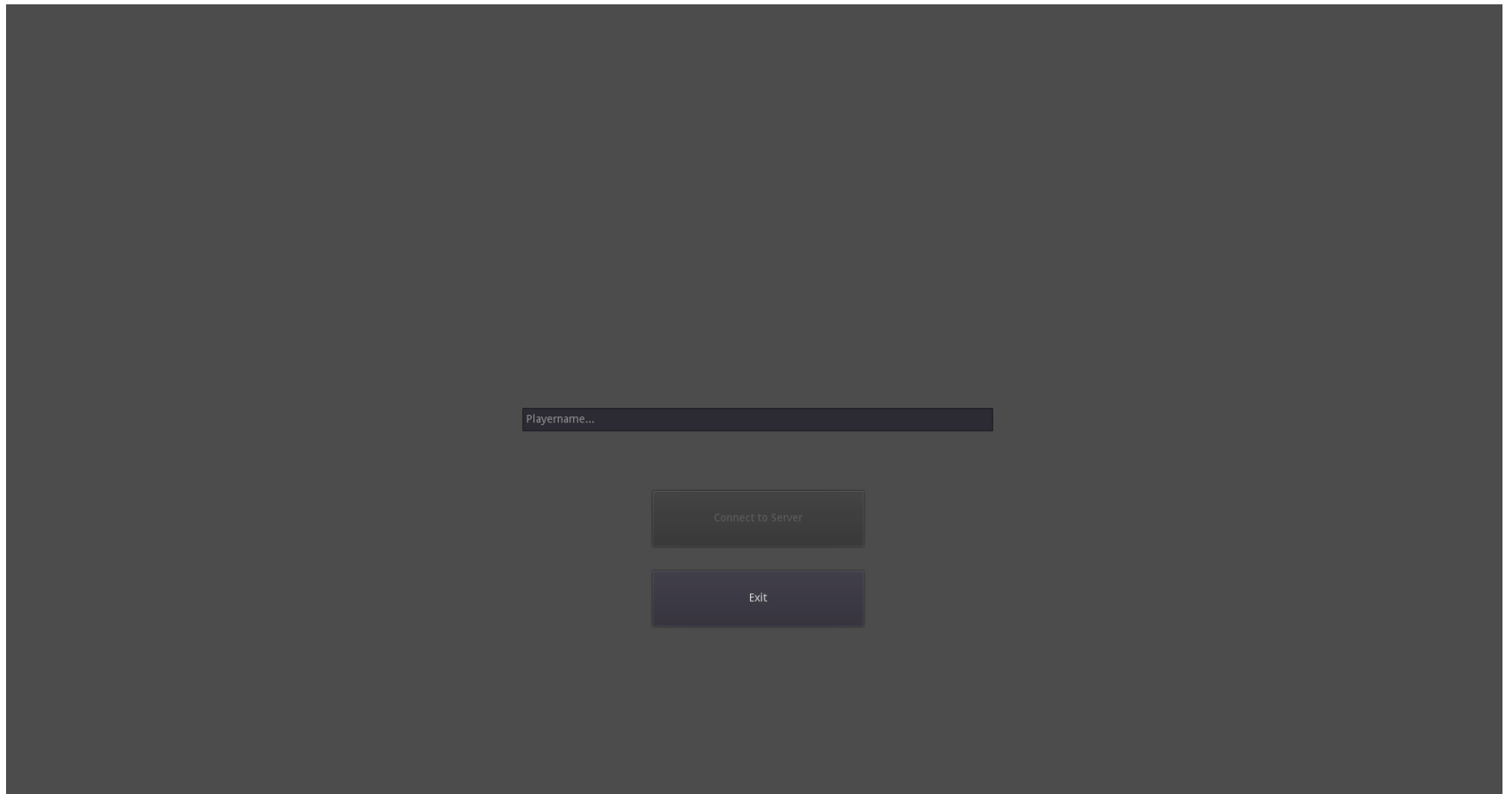
Solution

```
1 extends Node
2
3 const SERVER_IP = "127.0.0.1"
4 const SERVER_PORT = 3456
5 const MAX_PLAYERS = 1000
6
7 func _ready():
8     get_tree().connect("network_peer_connected", self, "_player_connected")
9     get_tree().connect("network_peer_disconnected", self, "_player_disconnected")
10    start_server()
11
12 func start_server():
13     print("Try to start the server.")
14     var peer = NetworkedMultiplayerENet.new()
15     var result = peer.create_server(SERVER_PORT, MAX_PLAYERS)
16
17     if result != OK:
18         print("Failed creating the server.")
19         return
20     else:
21         print("Created the server.")
22
23     get_tree().set_network_peer(peer)
24
25 func _player_connected(id):
26     print(str(id) + " connected to server.")
27     NetworkingSync._players_online.append(id)
28
29 func _player_disconnected(id):
```

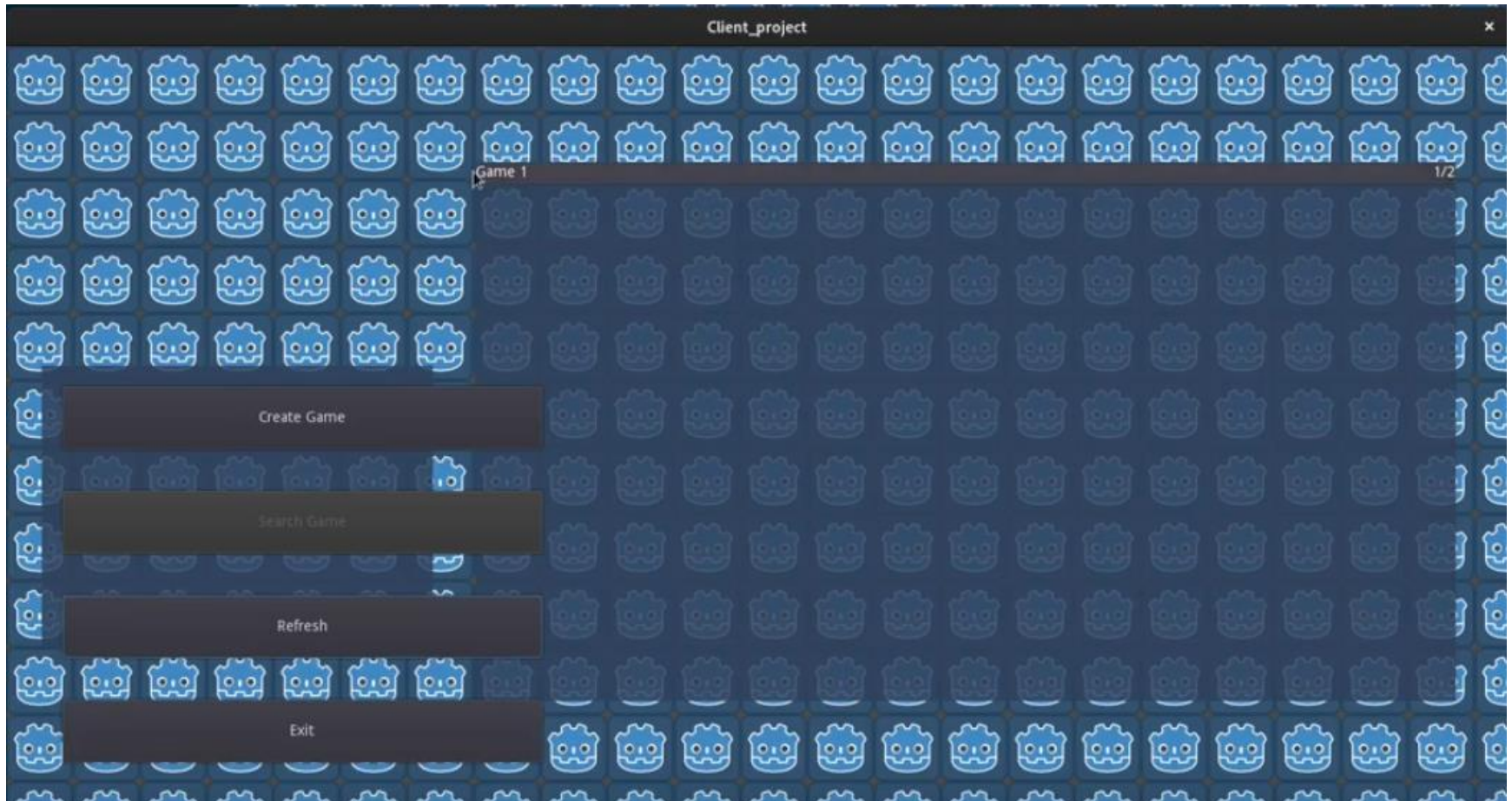
Solution

```
1 extends Node
2 # {game_id:[_game_id, _game_name, _max_players, _player_list]} # player_list is a dict
3 # {game_id:[_game_id, _game_name, _max_players, {player_id:[_player_name, _player_id, _game_id, _is_host]}}
4 var _open_games = {}
5
6 var _players_online = []
7
8 remote func get_open_games_from_server(id):
9     >| rpc_id(id, "update_open_games", _open_games)
10
11 remote func add_game_to_game_list(game_id, game_information, host_player):
12     >| _open_games[game_id] = game_information
13     >| add_player_to_open_game(game_id, host_player)
14     >| get_open_games_from_server(game_id)
15
16 remote func join_open_game(game_id, player_information):
17     >| add_player_to_open_game(game_id, player_information)
18
19 func add_player_to_open_game(game_id, player_information):
20     >| _open_games[game_id][3][player_information[1]] = player_information
21
22 remote func remove_player_from_open_game(game_id, player_id):
23     >| _open_games[game_id][3].erase(player_id)
24
25 remote func remove_game_from_game_list(game_id):
26     >| _open_games.erase(game_id)
27
```

Solution



Solution

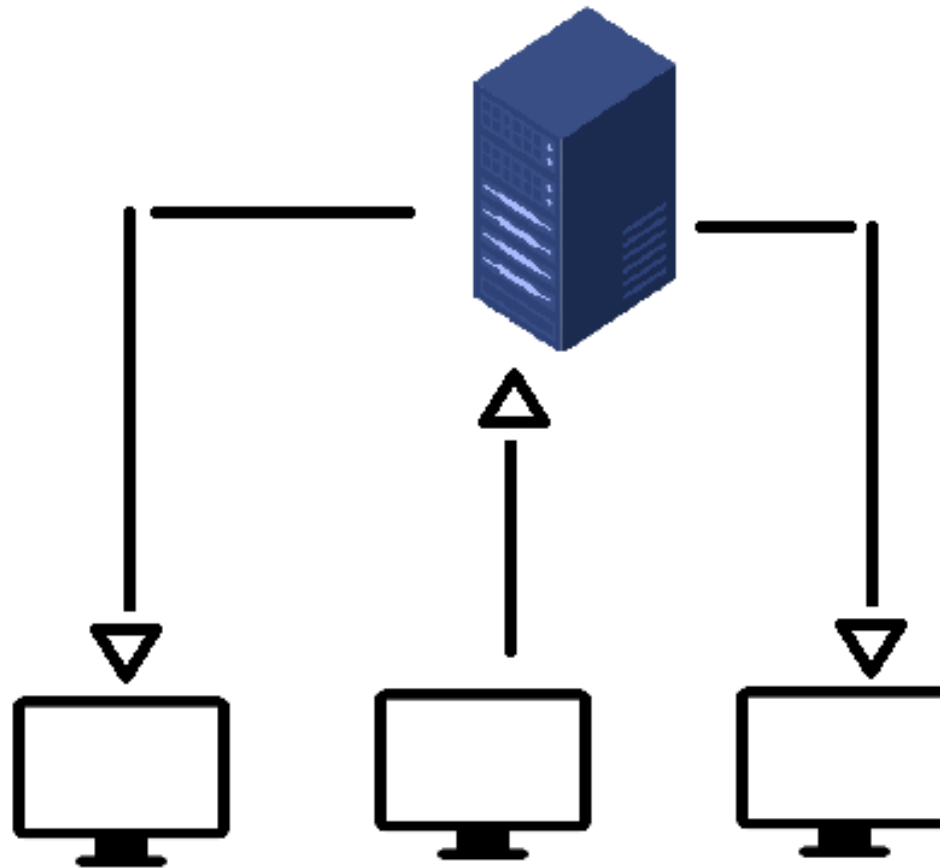


Solution

The image shows a dark-themed dialog box with a repeating pattern of small robot icons in the background. It contains three input fields and two buttons:

- Gamename:** A text input field that is currently empty.
- Max Players:** A text input field that is currently empty.
- Activity type:** A dropdown menu with the text "Code puzzle" and a downward-pointing triangle icon on the right.
- Buttons:** Two buttons at the bottom, labeled "Ok" and "Cancel".

Solution



Solution

Choose the right order of images

```
3 references
public abstract class Creature
{
    public const int baseHitPointsconst = 10;
    public const double baseDamageconst = 2.5;

    5 references
    public string Name { get; }
    8 references
    public double RockDamage { get; set; }
    8 references
    public double PaperDamage { get; set; }
    8 references
    public double ScissorDamage { get; set; }
    21 references
    public double Hitpoints { get; set; }
    19 references
    public int Level { get; set; }
    9 references
    public double BaseHitpoints { get; set; }

    2 references
    public Creature(string name)
    {
        this.Name = name;
        RockDamage = 2.5;
        PaperDamage = 2.5;
        ScissorDamage = 2.5;
        BaseHitpoints = baseHitPointsconst;
        Hitpoints = BaseHitpoints;
        Level = 1;
    }
}
```

```
3 references
class Hero : Creature, LvUpDamage
{
    4 references
    public const int baseLevelUpconst = 20;
    4 references
    public double Experience { get; set; } = 0;
    4 references
    public double ExperienceLevel { get; set; } = 20;

    1 reference
    public Hero(string name) : base(name)
    {
        BaseHitpoints = 15;
        Hitpoints = BaseHitpoints;
    }

    1 reference
    public double GainExperience(int monsterLevel)
    {
        double exp = (baseLevelUpconst/2 * monsterLevel);
        Experience += exp;
        return exp;
    }

    1 reference
    public void Heal()
    {
        Hitpoints = BaseHitpoints;
    }
}
```

```
2 references
class Game
{
    Random random = new Random();
    private Hero hero;
    2 references
    public Game()
    {
        NewGame();
        Console.WriteLine("Write down the name of" +
            " your character : ");
        hero = new Hero(Console.ReadLine());
    }

    1 reference
    public void NewGame()
    {
        Console.WriteLine("Hello, write down 'new game'" +
            " and we can start");
        while (Console.ReadLine() != "new game") ;
        Console.WriteLine("The game has begun!");
    }

    1 reference
    public Boolean WinGame()
    {
        if (hero.Level == 10)
        {
            Console.WriteLine("Your hero has reached" +
                " lvl 10! You have won!");
            Console.ReadKey();
            return true;
        }
        return false;
    }
}
```

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Game game = new Game();
        while (!game.WinGame())
        {
            if (!game.PlayGame())
            {
                game = new Game();
            }
        }
    }
}
```

SEND

Solution

- Scorelist with names exported

Problems & needs to be done

Problems & needs to be done

**This meme
had to be
censored**

Problems & needs to be done

Syncing all players
after choosing
answer

Problems & needs to be done

Syncing all players after choosing answer

Teacher user to manage activities in real time

Problems & needs to be done

Syncing all players after choosing answer

Teacher user to manage activities in real time

Other activities (backend already kinda done, mainly frontend)

Ideas for the future

- To conduct a research – use the app in an action

Ideas for the future

- To conduct a research – use the app in an action
- To modify the app to support more activities and more functionality

Ideas for the future

- To conduct a research – use the app in an action
- To modify the app to support more activities and more functionality
- To defend my bachelors thesis successfully

Ideas for the future

- To conduct a research – use the app in an action
- To modify the app to support more activities and more functionality
- To defend my bachelors thesis successfully
- To sleep at least for 12 hours

Questions?

Thanks for your attention

Mid shower when suddenly your bakalářka comes to your mind

