
Die Siedler von Catan

Gliederung

- Management
- Timeline
- Design
- Networking
- GameLogic
- Board
- Beispiele
- Fazit

Management

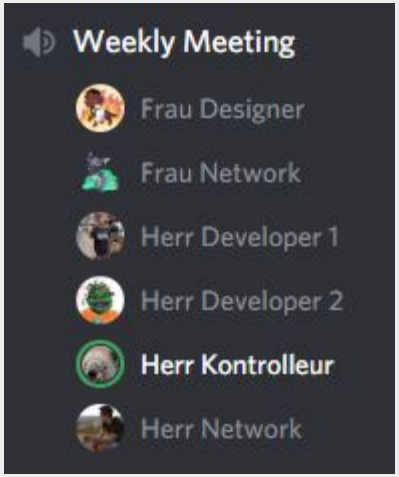
6er Team

1 Designer
2 Networker
3 Developer

- Lena
- Andrea, Simon
- Marco, Maxi, Timo

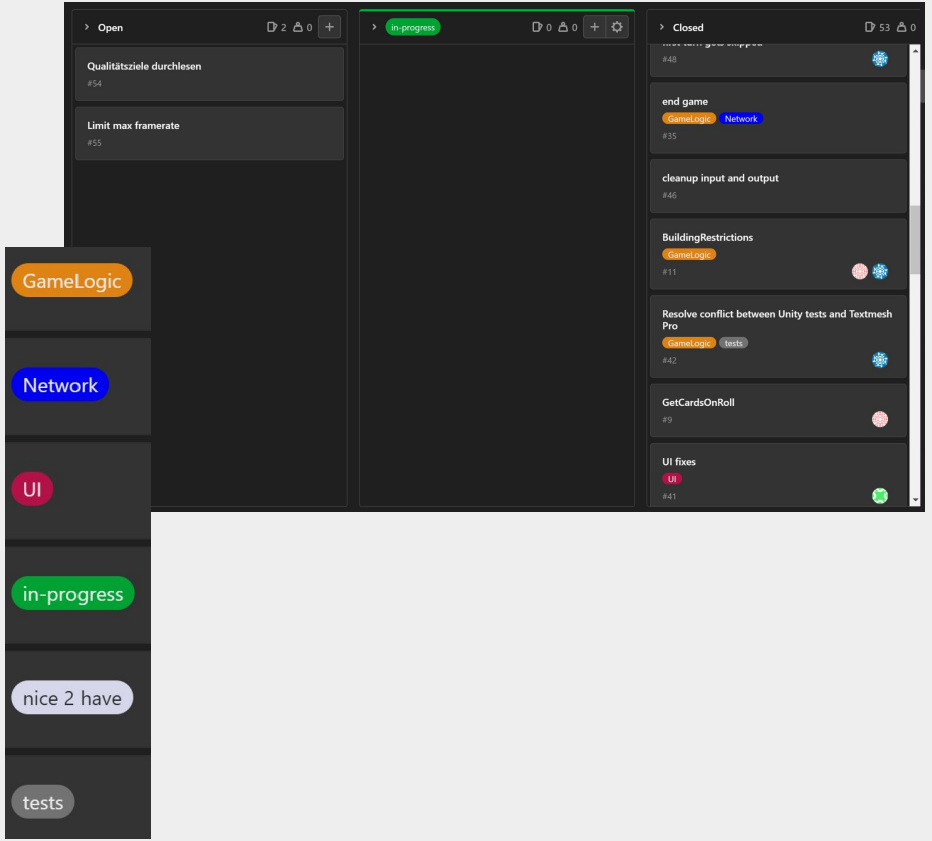
Meetings

Mittwochs 16:00
Donnerstags 8:15



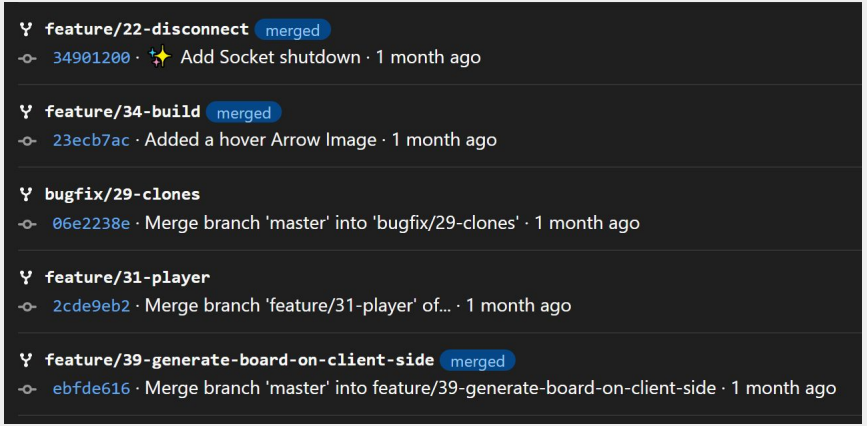
Issues Issues Issues

open ➔ in-progress ➔ closed



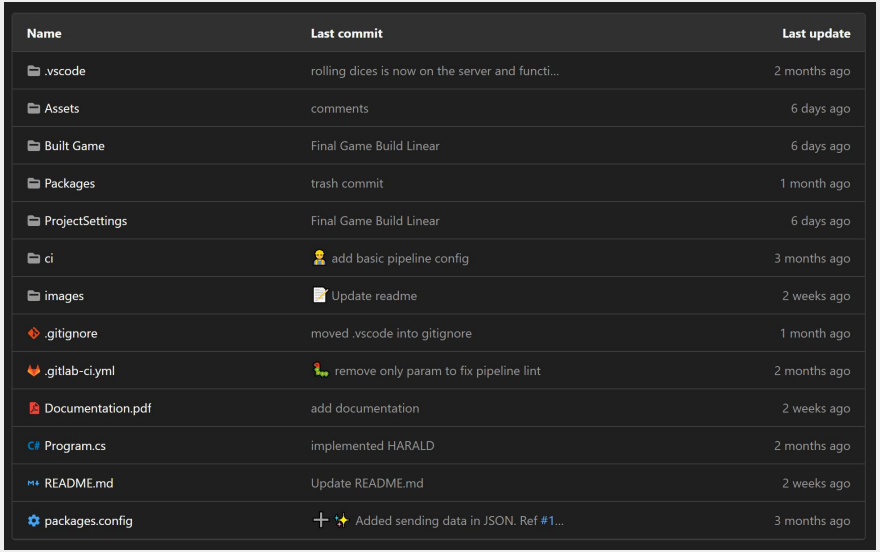
Branching

feature/bugfix/tests/...



Clean Repo

- klare Ordnerstruktur
- kein Müll
- README für User



Name	Last commit	Last update
.vscode	rolling dices is now on the server and functi...	2 months ago
Assets	comments	6 days ago
Built Game	Final Game Build Linear	6 days ago
Packages	trash commit	1 month ago
ProjectSettings	Final Game Build Linear	6 days ago
ci	add basic pipeline config	3 months ago
images	Update readme	2 weeks ago
.gitignore	moved .vscode into gitignore	1 month ago
.gitlab-ci.yml	remove only param to fix pipeline lint	2 months ago
Documentation.pdf	add documentation	2 weeks ago
Program.cs	implemented HARALD	2 months ago
README.md	Update README.md	2 weeks ago
packages.config	Added sending data in JSON. Ref #1...	3 months ago

Timeline

Basic Design
Network
Full Design
Game Logic

Anfang April

Timeline

Basic Design
Network
Full Design
Game Logic

Anfang Mai

```
private static void sendRequest()
{
    Console.Write("Send a request: (get status)");
    //string request = Console.ReadLine();
    string request = "Hallo Welt";

    byte[] buffer = Encoding.ASCII.GetBytes(request);
    clientSocket.Send(buffer, 0, buffer.Length, SocketFlags.None);

    //TODO: send game data and messages -> may send JSON
}
```

Timeline

Basic Design
Network
Full Design
Game Logic

Ende Mai

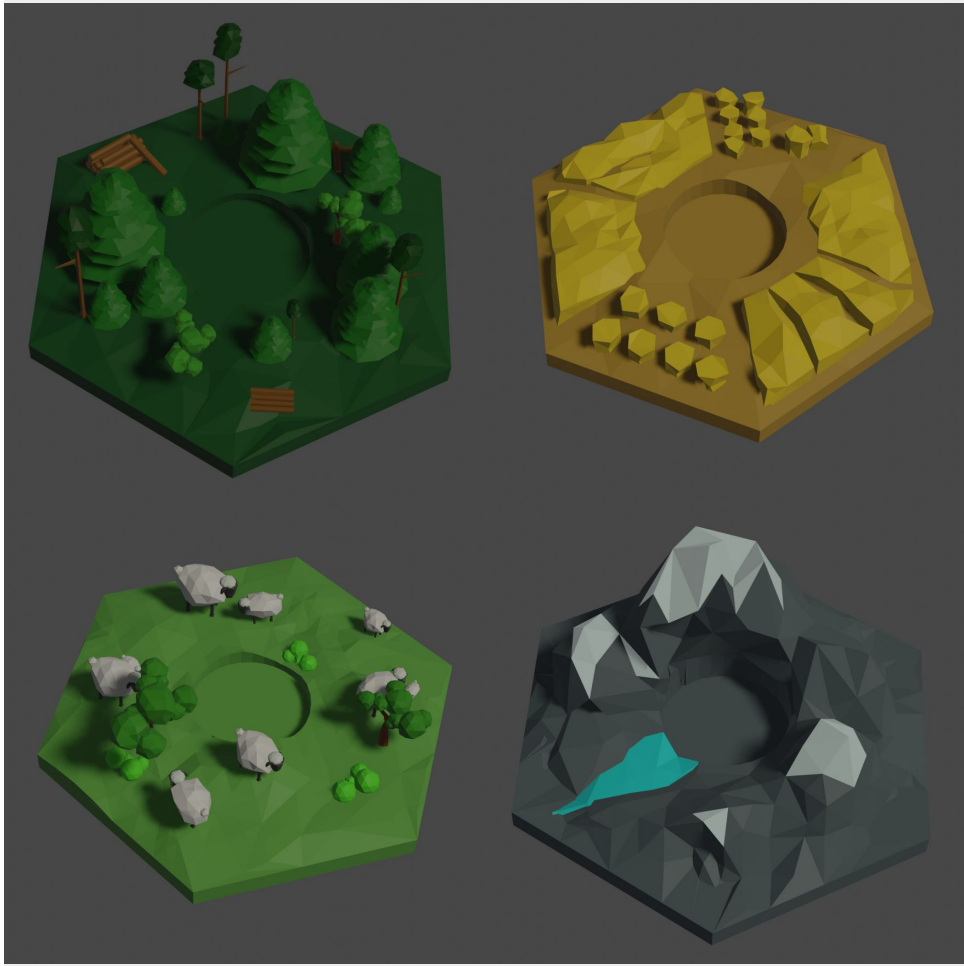
Timeline

Basic Design
Network
Full Design
Game Logic

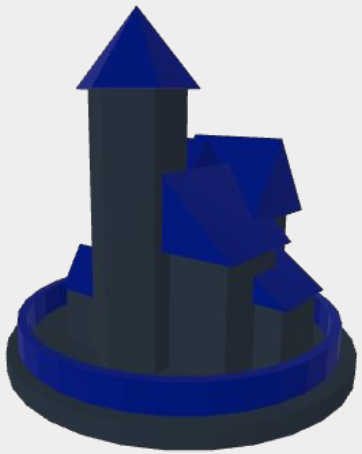
Anfang Juni

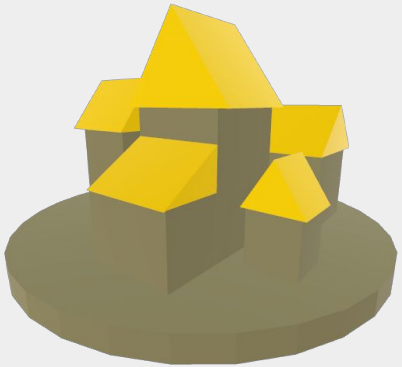
Design

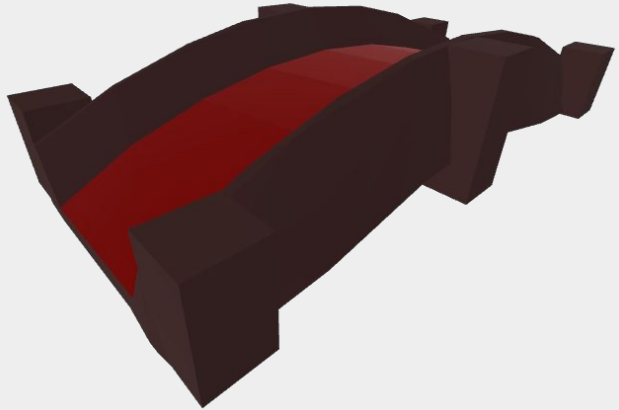
Hexagons



Buildings

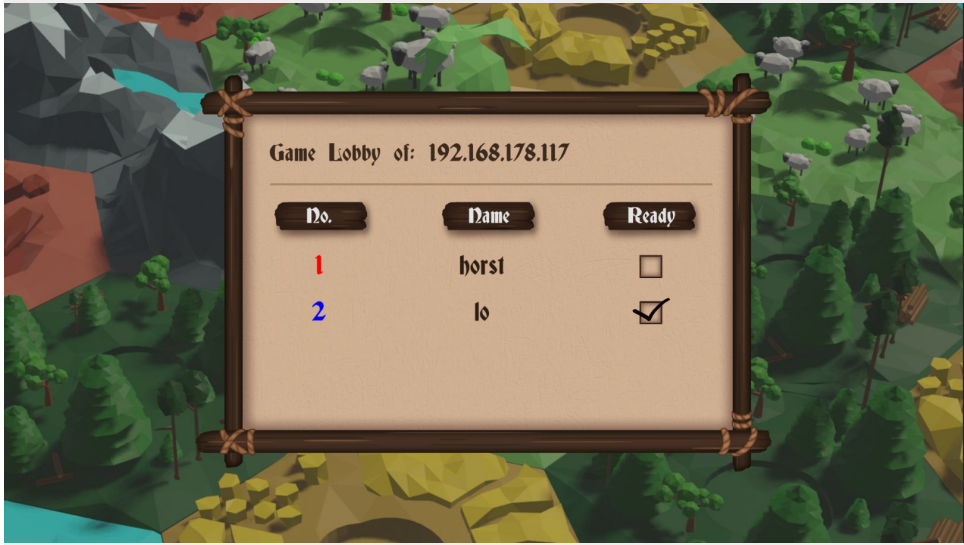






UI-Elements

Lobby



Ingame

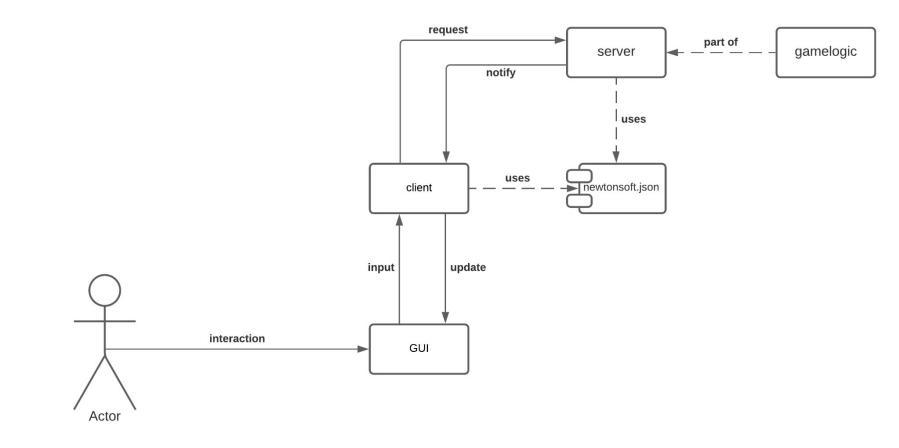

horst (you)
2 10 0
Io
2 7 0
trade
15 build street
3 build village
4 build city
DevCards
show costs
next player
3 2 0 2 3

Pop-up


horst (you)
2 7 1
lo
2 7 0
Builiding costs
street
0 VP
village
1 VP
city
2 VP
development card
? VP
close
trade
13 build street
3 build village
4 build city
DevCards
show costs
next player
2 2 0 1 2

Architektur

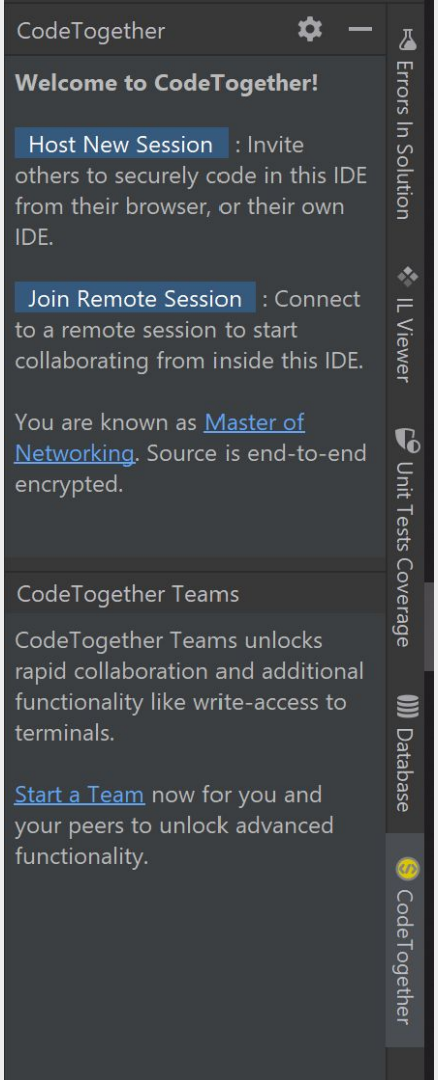
grober Überblick

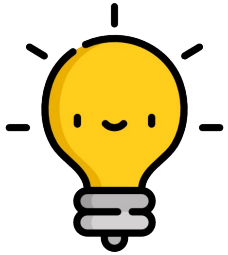


Networking

Entwicklung mit CodeTogether

Simon und Andrea





Networking V1

Intelligente Clients
Echo Server

- erste Idee
- Client besitzt Spiellogik
- Server leitet nur weiter
- erschien simpel

LightBulb von https://www.flaticon.com/

Erste Implementation (Strings versenden)

```
byte[] currentBuffer = new byte[recievedByteLengh];
Array.Copy(buffer, currentBuffer, recievedByteLengh); //to remove the protruding zeros from buffer
string incomingDataString = Encoding.ASCII.GetString(currentBuffer);
Console.WriteLine("Received Text: " + incomingDataString);
//todo: handle incomingDataString

if (incomingDataString.ToLower().Equals("get status"))
{
    string dataString = $"Current status: \n connected clients: {clientSockets.Count} \n current buffer size: {BUFFER_SIZE}";
    byte[] dataToSend = Encoding.ASCII.GetBytes(dataString);
    currentClientSocket.Send(dataToSend);
    Console.WriteLine("Current status was requested and sent.");
} else if (incomingDataString.ToLower().Equals("exit")) // Client wants to exit gracefully
```

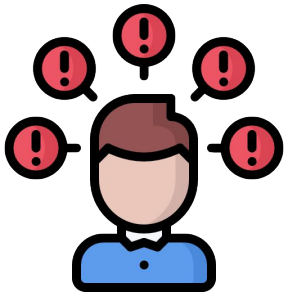
Serverseitiger String empfang

Clientseitig String versenden

```
private static void sendRequest()
{
    Console.Write("Send a request: (get status)");
    //string request = Console.ReadLine();
    string request = "Hallo Welt";

    byte[] buffer = Encoding.ASCII.GetBytes(request);
    clientSocket.Send(buffer, 0, buffer.Length, SocketFlags.None);

    //TODO: send game data and messages -> may send JSON
}
```

Probleme mit V1

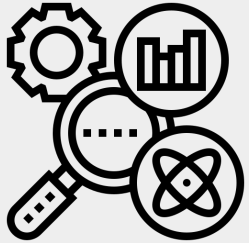
Der Hirnknoten



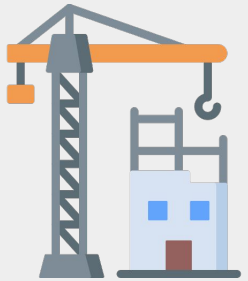
- Client weiß zu viel
- Cheater???
- Woher soll der Server wissen ob das jetzt für alle ist oder nicht?
- Synchronisation

Networking V2

das Umplanen
und der Umbau

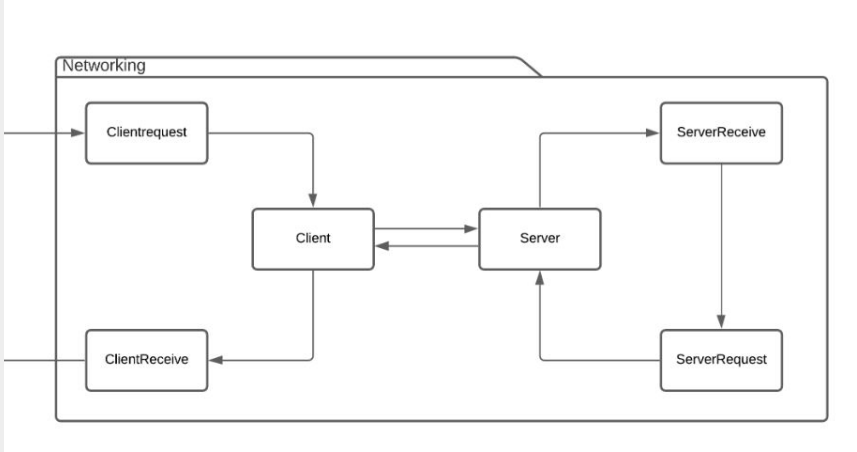


- Recherche
- alten Code prüfen
- alten Code umbauen



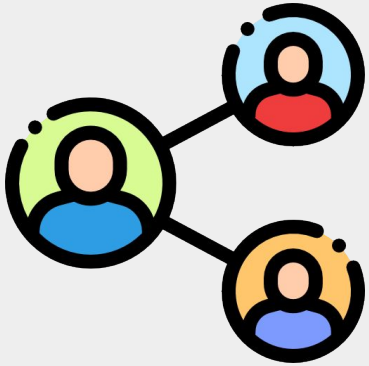
Neue Architektur

- Server beherbergt Logik
- Client kann nur Funktionalität anfragen

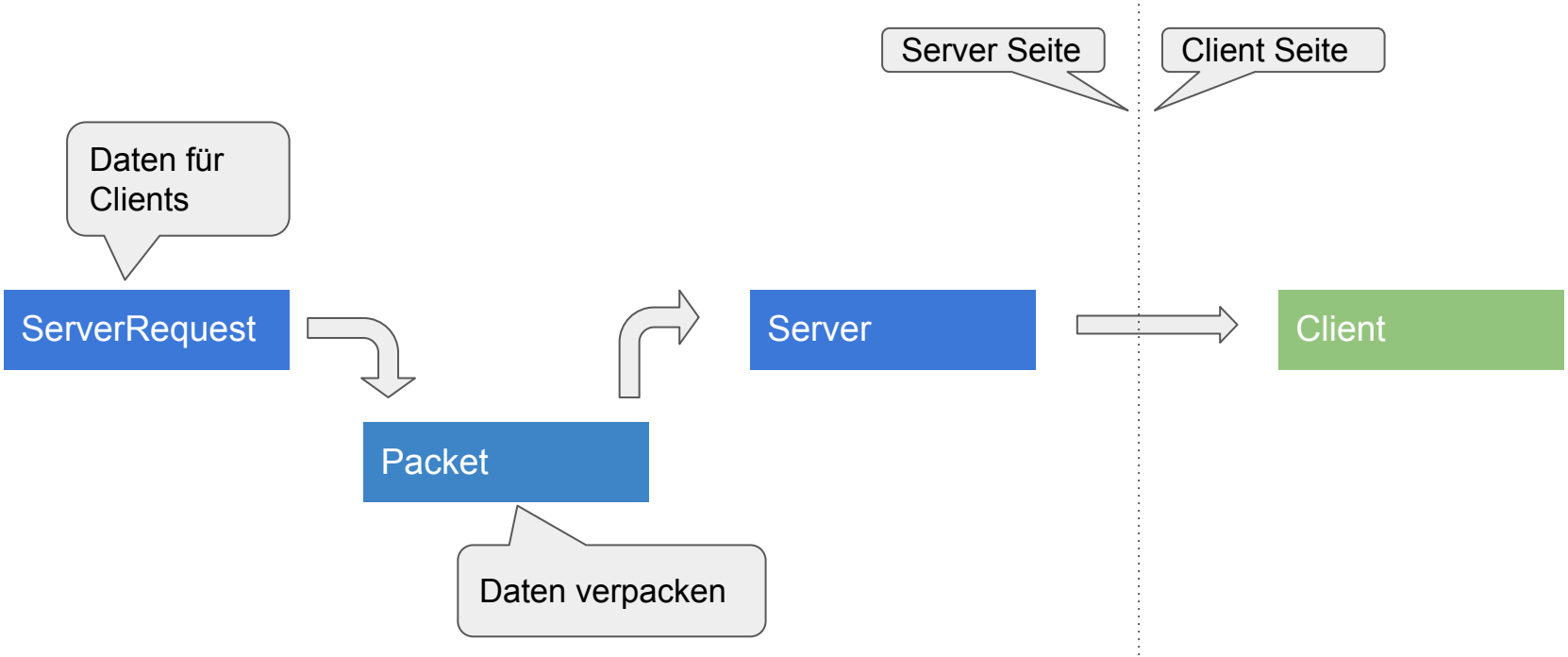


Kommunikation

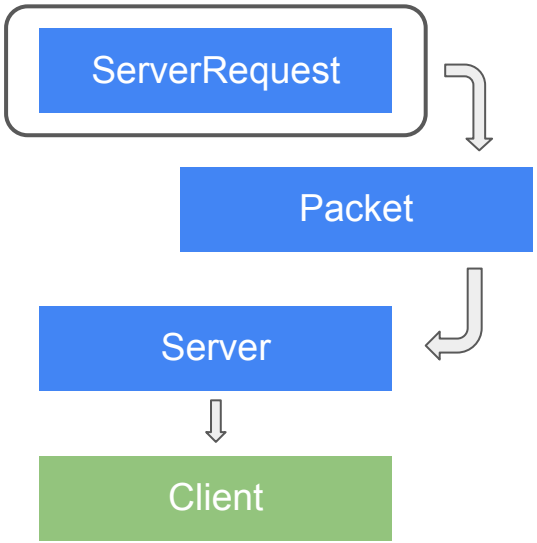
Server ➔ Client



Zusammenspiel



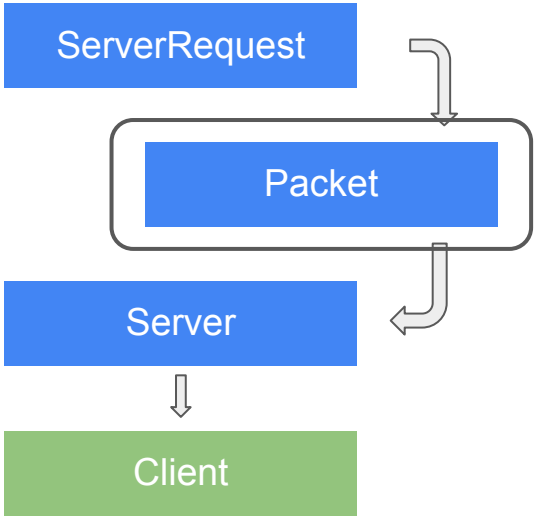
ServerRequest



Informationen zum verschicken

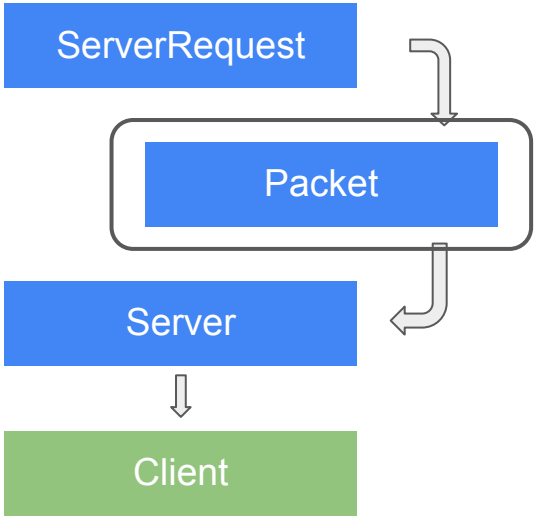
```
public void notifyClientJoined(ArrayList playerInformation, string lobbyIP)
{
    Packet packet = new Packet();
    packet.type = (int)COMMUNICATION_METHODS.HANDLE_CLIENT_JOINED;
    packet.lobbyContent = playerInformation;
    packet.lobbyIP = lobbyIP;

    // send to all
    Server.sendDataToAll(packet);
    Debug.Log( message: "SERVER: Client should expect a NotifyClientJoined package Type 10");
}
```

Packet




```
public void notifyClientJoined(ArrayList playerInformation, string lobbyIP)
{
    Packet packet = new Packet();
    packet.type = (int)COMMUNICATION_METHODS.HANDLE_CLIENT_JOINED;
    packet.lobbyContent = playerInformation;
    packet.lobbyIP = lobbyIP;

    // send to all
    Server.sendDataToAll(packet);
    Debug.Log( message: "SERVER: Client should expect a NotifyClientJoined package Type 10");
}
```

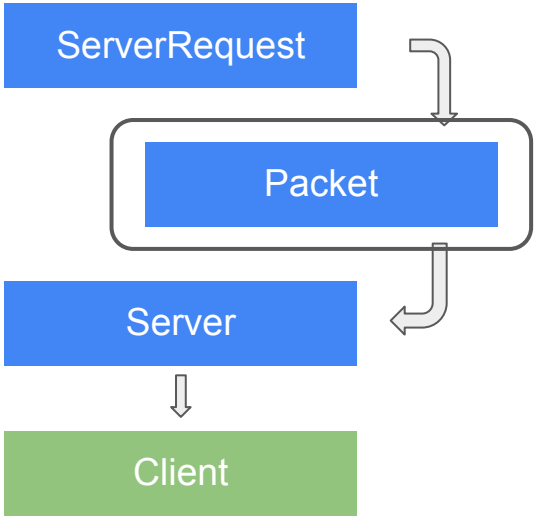
Packet




```
public class Packet
{
    Frequently called  48 usages
    public int type { get; set; } // what method needs to be called? -> set default to -1 to prevent wrong messages
    15 usages
    public string playerName { get; set; }
    5 usages
    public PLAYERCOLOR playerColor { get; set; } // [r,g,b,a]
    44 usages
    public int myPlayerID { get; set; } // player ID of the client who receives the packet

    8 usages
    public int? currentPlayerID { get; set; } // ID of the current player
    5 usages
    public int? previousPlayerID { get; set; } // ID of the previous player

    8 usages
    public bool                                          y only)

    3 usages
    public string
```

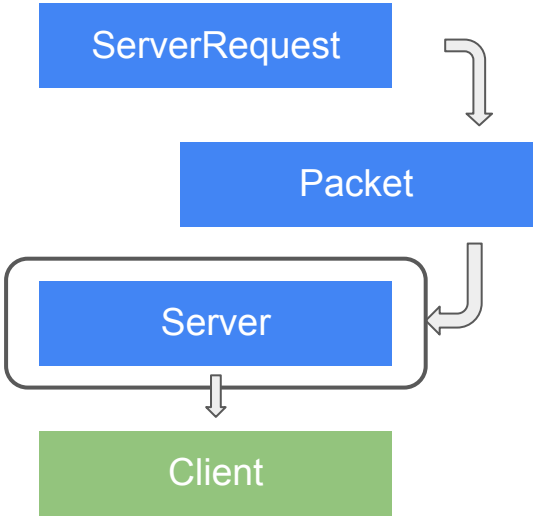
Nullable Int ➔ Platz sparen

Packet



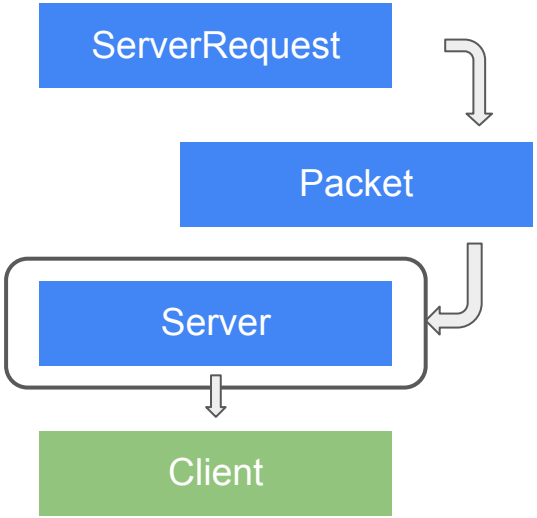

```
json = JsonConvert.SerializeObject(packet,
    new JsonSerializerSettings { NullValueHandling = NullValueHandling.Ignore });
```

Server




```
public void notifyClientJoined(ArrayList playerInformation, string lobbyIP)
{
    Packet packet = new Packet();
    packet.type = (int)COMMUNICATION_METHODS.HANDLE_CLIENT_JOINED;
    packet.lobbyContent = playerInformation;
    packet.lobbyIP = lobbyIP;

    // send to all
    Server.sendDataToAll(packet);
    Debug.Log( message: "SERVER: Client should expect a NotifyClientJoined package Type 10");
}
```

Server




```
public static void sendDataToOne(int playerID, Packet data)
{
    data.myPlayerID = playerID;
    string dataString = PacketSerializer.objectToJsonString(data);
    byte[] dataToSend = Encoding.ASCII.GetBytes(dataString);
    socketPlayerData[playerID].BeginSend(dataToSend, offset: 0, size: dataToSend.Length, SocketFlags.None, sendCallback, serverSocket);
    Thread.Sleep( millisecondsTimeout: 50);
}
```

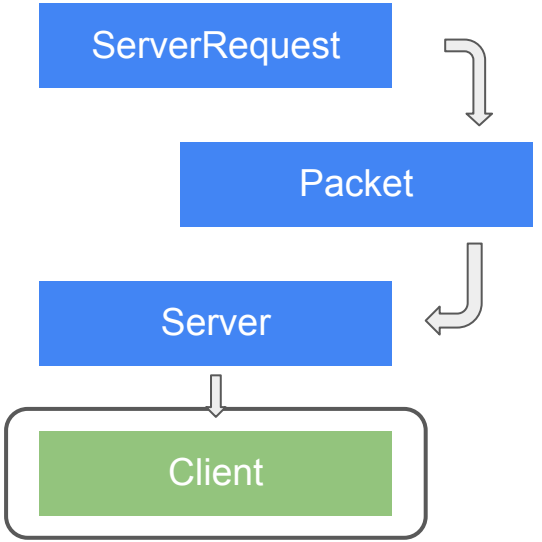
```
public static void sendDataToAll(Packet data)
{
    foreach (int id in socketPlayerData.Keys)
    {
        data.myPlayerID = id;
        string dataString = PacketSerializer.objectToJsonString(data);
        byte[] dataToSend = Encoding.ASCII.GetBytes(dataString);
        socketPlayerData[id].BeginSend(dataToSend, offset: 0, size: dataToSend.Length, SocketFlags.None, sendCallback, serverSocket);
    }
    Thread.Sleep( millisecondsTimeout: 50);
}
```

String

byte Array

versenden

Client




```
private static void delegateIncomingDataToMethods(Packet incomingData)
{
    try
    {
        switch (incomingData.type)
        {
            case (int)COMMUNICATION_METHODS.HANDLE_CLIENT_JOINED:
                ThreadManager.executeOnMainThread(() =>
                {
                    _clientReceive.handleClientJoined(incomingData);
                });
                break;

            case (int)COMMUNICATION_METHODS.HANDLE_GAMESTART_INITIALIZE:
                ThreadManager.executeOnMainThread(() =>
                {
                    _clientReceive.handleGameStartInitialize(incomingData);
                });
                break;

            case (int)COMMUNICATION_METHODS.HANDLE_PLAYER_READY_NOTIFICATION:
                ThreadManager.executeOnMainThread(() =>
                {
                    _clientReceive.handlePlayerReadyNotification(incomingData);
                });
                break;
```

Remember?

```
Packet packet = new Packet();
packet.type = (int)COMMUNICATION_METHODS.HANDLE_CLIENT_JOINED;
```

ThreadManager

```
public static void executeOnMainThread(Action action)
{
    if (action == null)
    {
        Debug.LogWarning( message: "CLIENT: Action to execute on main thread is null!");
        return;
    }
    // secure that execOnMainThreadList is not read at the same time.
    lock (execOnMainThreadList)
    {
        execOnMainThreadList.Add(action);
        actionsToExec = true;
    }
}
```

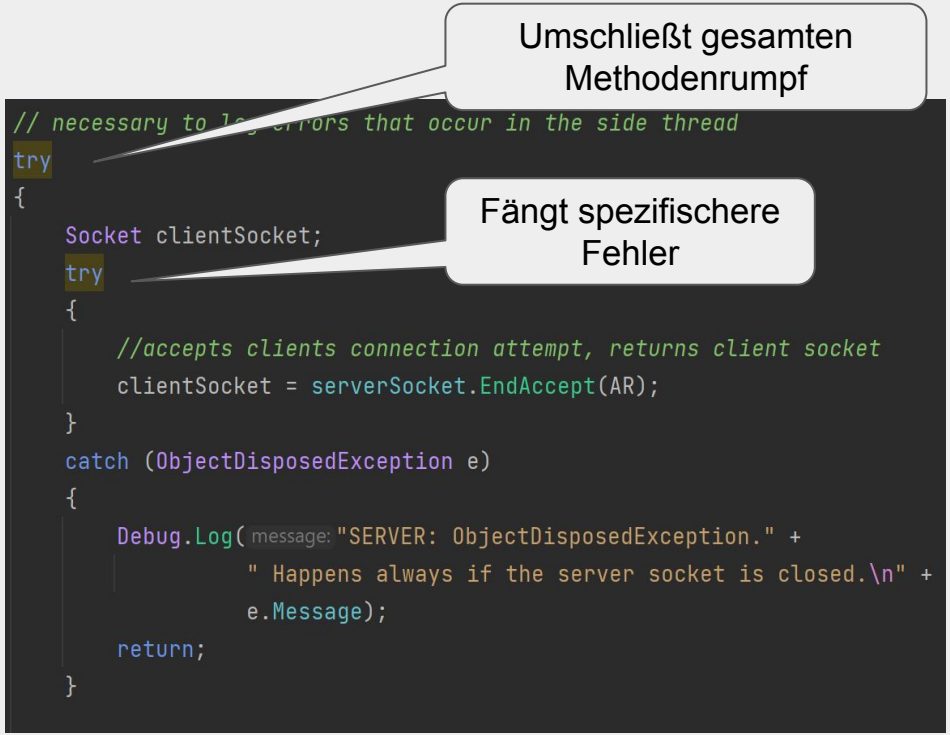
Von jedem thread aufrufbar

Auf Unity’s main thread ausführen

```
foreach (Action action in execOnMainThreadListCopied)
{
    action();
}
```

TryCatch

Unity und sein main Thread



GameLogic

Regeln

Spielbrett
Startphase
Bauen
Traden
Development Cards

- Wüsten, Ressourcen, Häfen

Regeln

Spielbrett
Startphase
Bauen
Traden
Development Cards

```
// Begin next round
if (!inGameStartupPhase)
{
    changeCurrentPlayer(clientPacket, currentPlayer);
    Debug.Log("SERVER: Current Player index: " + currentPlayer);
    updateRepPlayers();
    updateOwnPlayer(currentPlayer);
    handleBeginRound(clientPacket);
}
```

Regeln

Spielbrett
Startphase
Bauen
Traden
Development Cards

- Straßen, Dörfer, Städte

Regeln

Spielbrett
Startphase
Bauen
Traden
Development Cards

- 4:1 / 3:1 / 2:1

Regeln

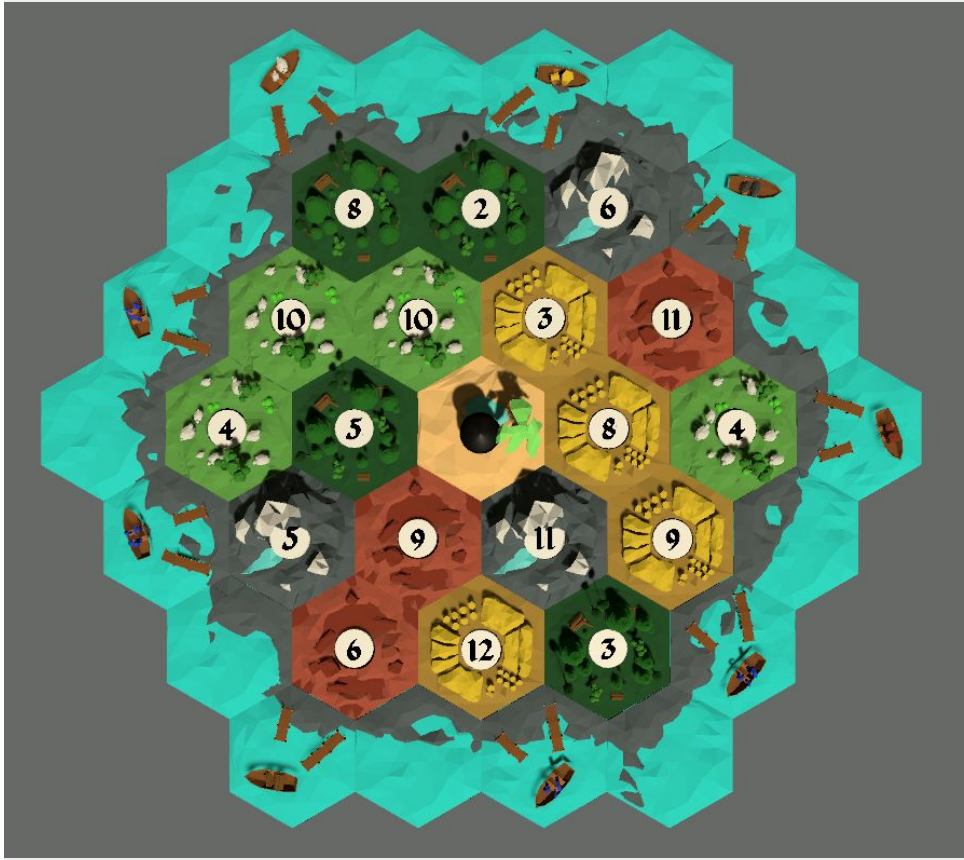
Spielbrett
Startphase
Bauen
Traden
Development Cards

- Vorgesetzte Kartenanzahl

Board

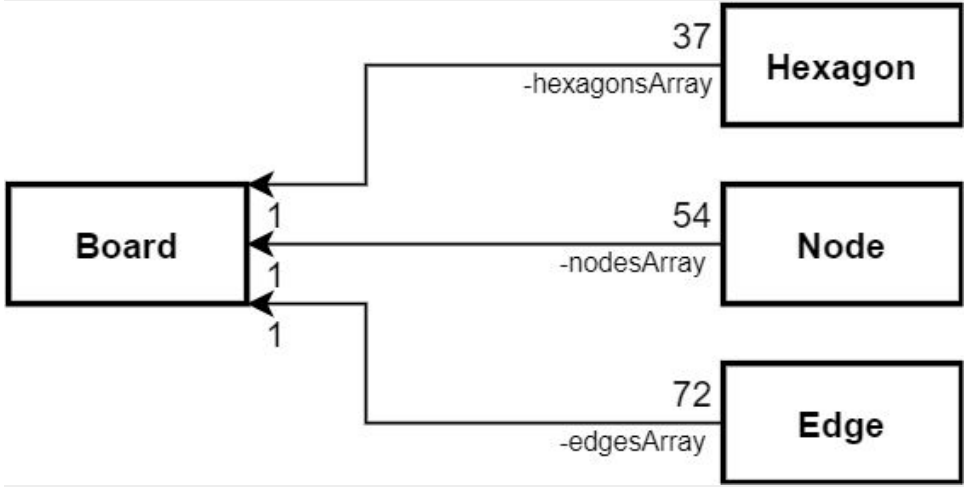
Das Spielbrett

- Zufällig angeordnete Hexagons & Zahlenfeldern
- Feste Positionen für Wüste, Häfen und Landfelder
- Feste Bauslots



Aufbau des Spielbretts

- Die Klasse ‘Board’ hält alle für das Spielbrett nötigen Funktionen und Referenzen
- Hexagons, Nodes und Edges in separaten Arrays



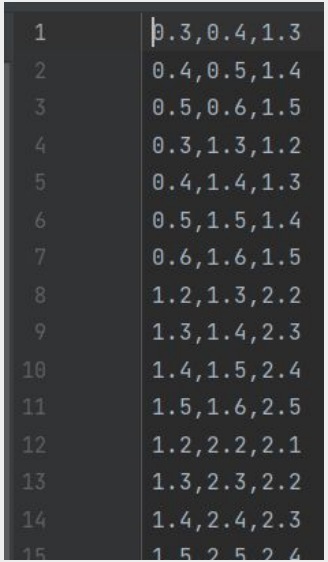
Initialisierung des Boards

- Initialisierung der Kindelemente
- Überprüfen der Platzierungsvorgaben
- Zuweisung der Nachbarelemente

```
public Board()
{
    initializeNodes();
    initializeEdges();
    initializeHexagons();
    checkPlacementConstraints();
    assignNeighborsToHexagons();
    assignNeighborsToNodes();
    assignNeighborsToEdges();
}
```

Nachbar-elemente

- Hexagons, Nodes und Edges müssen ihre Nachbarn kennen
- Positionen immer dieselben



Platzierung der Hexagons

- Werden in 2D-Array gespeichert
- Boardconfig gibt vor, wo welche Art von Hexagon liegt

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5	q = 6
r = 0	(null)	(null)	(null)	3, 0	4, 0	5, 0	6, 0
r = 1	(null)	(null)	2, 1	3, 1	4, 1	5, 1	6, 1
r = 2	(null)	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2
r = 3	0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3
r = 4	0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	(null)
r = 5	0, 5	1, 5	2, 5	3, 5	4, 5	(null)	(null)
r = 6	0, 6	1, 6	2, 6	3, 6	(null)	(null)	(null)

```
private readonly int[][] boardConfig = {
    new[] {0,0,0,4,1,4,1},
    new[]  {0,0,1,2,2,2,4},
    new[]   {0,4,2,2,2,2,1},
    new[]    {1,2,2,3,2,2,4},
    new[]     {4,2,2,2,2,1,0},
    new[]      {1,2,2,2,4,0,0},
    new[]       {4,1,4,1,0,0,0}
};
```

Zufällige Anordnung

- Zuweisung der Zufallszahlen bei Initialisierung
- Überprüfen der Platzierungsvorgaben

```
private void initializeHexagons()
{
    Stack<HEXAGON_TYPE> landStack = createRandomHexagonStackFromArray(
    Stack<HEXAGON_TYPE> portStack = createRandomHexagonStackFromArray(
    numberStack = createRandomHexagonNumberStack(availableNumbers);
```

```
/// <summary>
/// Checks if Hexagons with a fieldnumber of 6 or 8 are adjacent
/// </summary>
2 references
private void checkPlacementConstraints()
{
```

Umsortieren der Hexagons

- Ermitteln einer geeigneten Position
- Geeignet, wenn Nachbarfeld keine 6 oder 8

```
/// <summary>
/// checks the hexagons array for a suitable position where none
/// of the adjacent hexagons has a 6 or 8 as fielnumber.
/// </summary>
/// <returns>an int array with the first occouring coordinates of a
/// suitable position or empty array if no suitable position is found
/// </returns>
1 reference
private int[] findSuitablePos()
{
```

Weitere Funktionalitäten

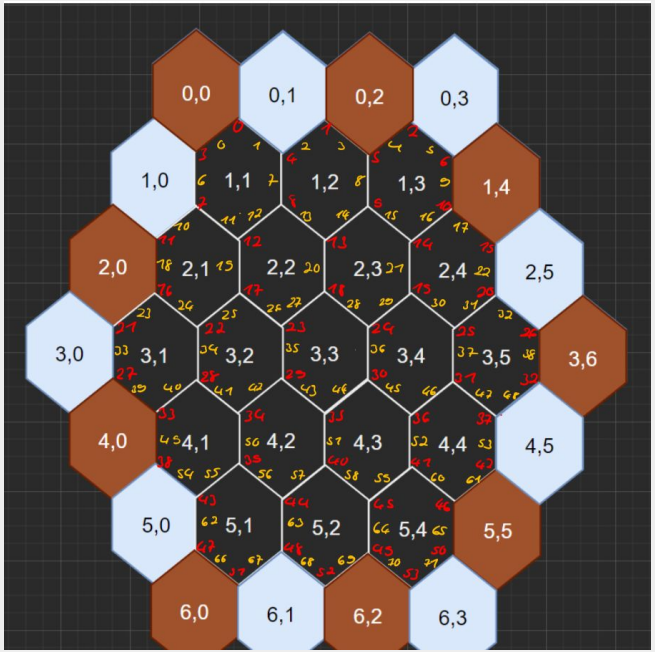
- Überprüfen der Platzierungsvorgaben für Gebäude
- Platzierung von Gebäuden
- Austeilen von Ressourcen

BoardGenerator

- Clientseitig
- Regelt die Darstellung des Boards in Unity
- Instanziiert Prefabs

gemachte Fehler

- Ursprünglich 3 Offset-Arrays nötig
- Aufwendige Array-Struktur (z.B. Jagged Array)




```
---[obere Hälfte]--- | -------[Mitte]-----|---[untere Hälfte]---
y |-1,-1, 0, 1, 1, 0 | -1,-1, 0, 1, 1, 0  | -1,-1, 0, 1, 1, 0
x | 0,-1,-1, 0, 1, 1 |  0,-1,-1,-1, 0, 1  |  1, 0,-1,-1, 0, 1
```

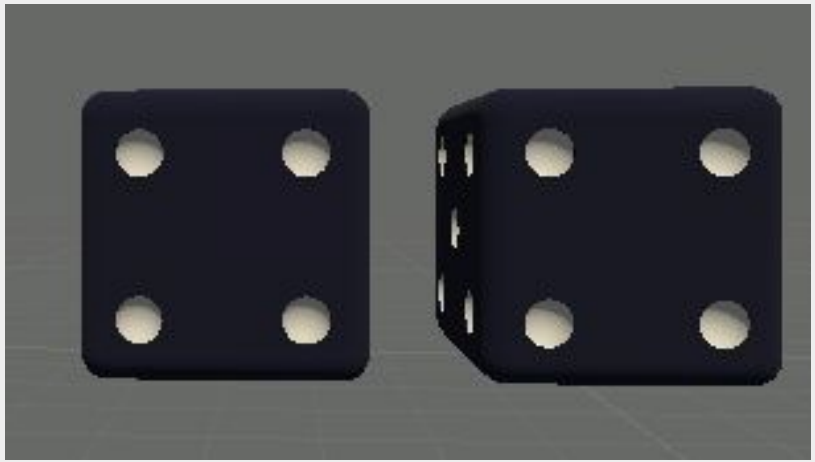
mögliche Verbesserungen

- FindSuitablePos Method sollte alle passenden Positionen finden und zufällig auswählen
- Intelligentere Nummernzuweisung von Beginn

Beispiele

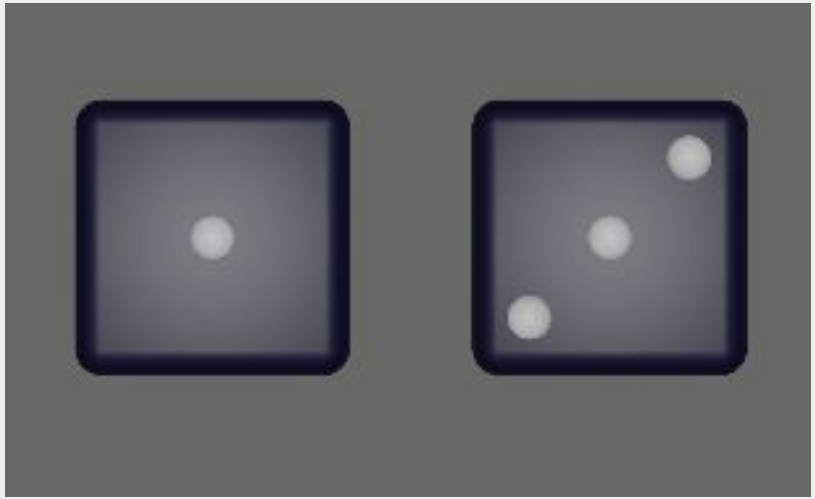
Würfeln

Ansatz V1: 3D



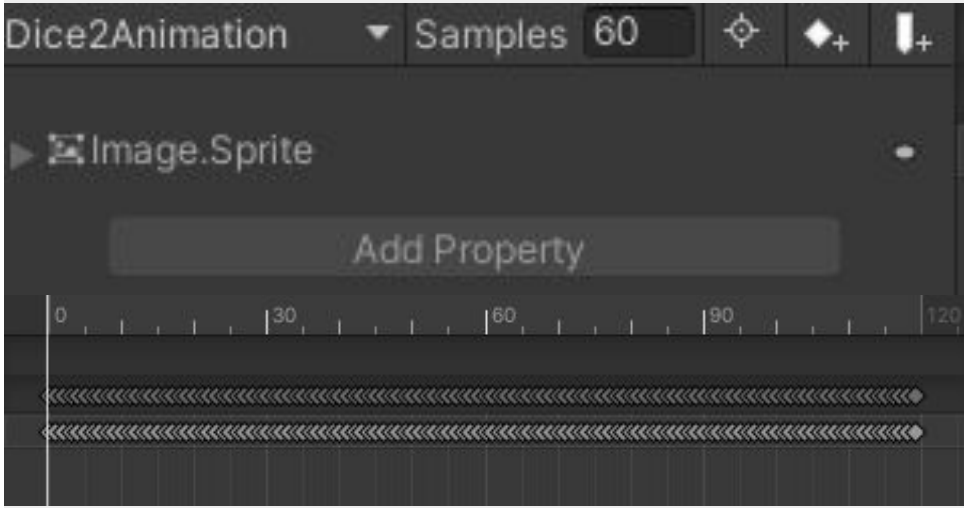
Würfeln

Ansatz V2: Sprites



Würfeln

Ansatz V2: Animation



Würfeln

Ansatz V2: Coroutine

```
public IEnumerator TimeYield()
{
    Debug.Log("CLIENT: TimeYield Function is called");
    yield return new WaitForSeconds(0.5f);
    Debug.Log("CLIENT: Waited 2 Seconds");
    Dice1.updateDiceNumber(diceNumbers[0]);
    yield return new WaitForSeconds(0);
    Dice2.updateDiceNumber(diceNumbers[1]);
}
```

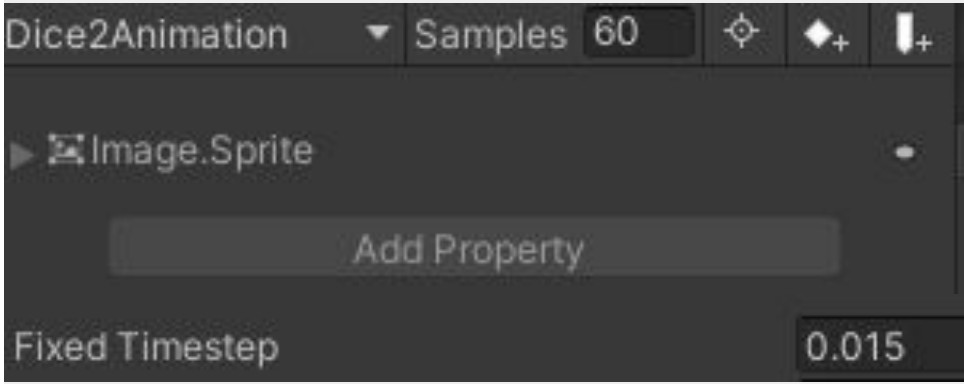
Würfeln

Ansatz V2: Coroutine

```
private String[] diceNumberStrings = {
    "0001 (UnityEngine.Sprite)",
    "0020 (UnityEngine.Sprite)",
    "0040 (UnityEngine.Sprite)",
    "0060 (UnityEngine.Sprite)",
    "0080 (UnityEngine.Sprite)",
    "0100 (UnityEngine.Sprite)" };
void FixedUpdate()
{
    if (diceAnimator.enabled)
    {
        if (diceNumber == diceImage.sprite.ToString())
        {
            diceAnimator.enabled = false;
            diceNumber = "";
            Debug.Log("SERVER: Animating: " + diceAnimator.enabled);
        }
    }
}
```

Würfeln

Fixed Framerate

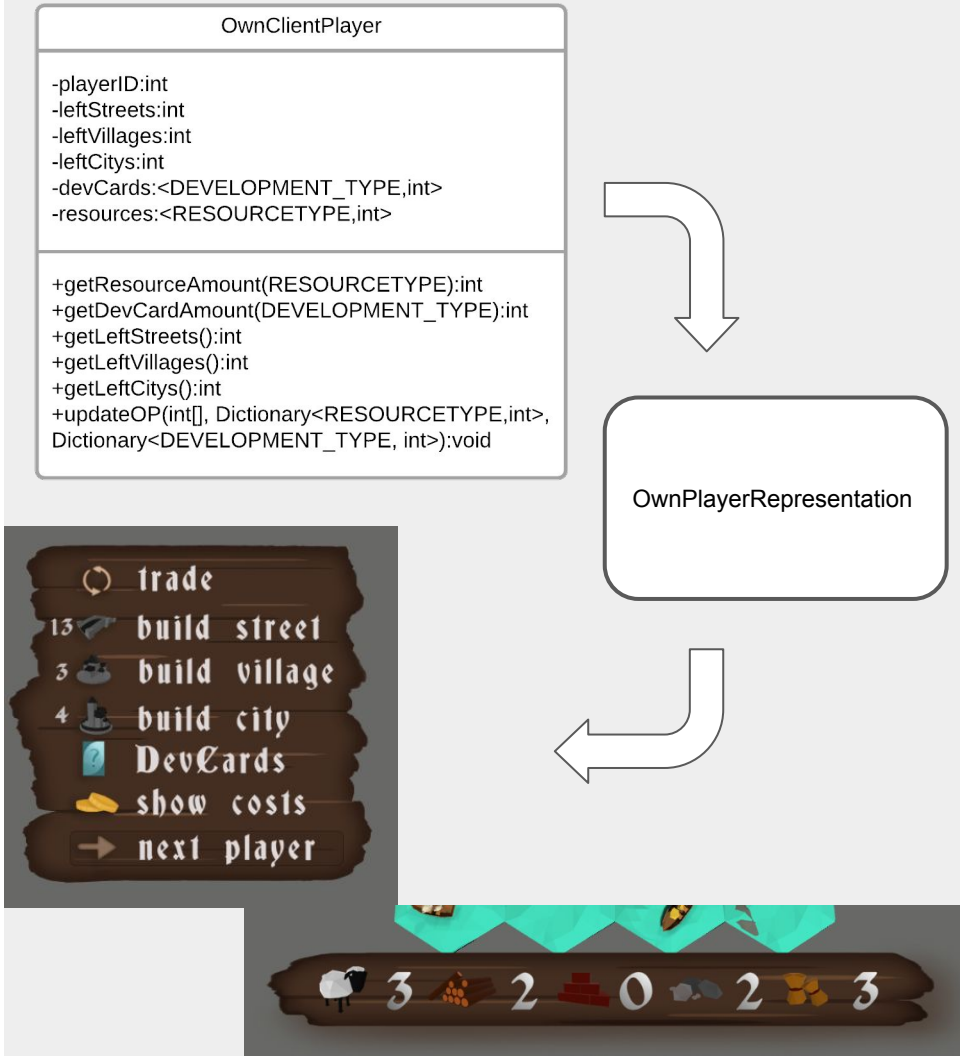


Player

ServerPlayer

Player

OwnClientPlayer



Player

RepresentativePlayer

PlayerRepresentation

RepresentativePLayer
-playerID:int -playerName:int -playerColor:Color -victoryPoints:int -totalResourceAmount:int -devCardAmount:int
+getPlayerColor():Color +getPlayerName():String +getVictoryPoints():int +getTotalResourceAmount():int +getDevCardAmount():int +getPlayerID():int +updateNumbers(int[] updates):void

Fazit

Fazit

- Neue Umgebung lernen
- Regelmäßige Absprachen
- Zuständigkeiten
- Deadlines einhalten
- Rad nicht neu erfinden (Server)
- Bei Problemen direkt melden
- Datendurchsatz