Stonecap Mountain

Beta Release

Torii Greiskalns

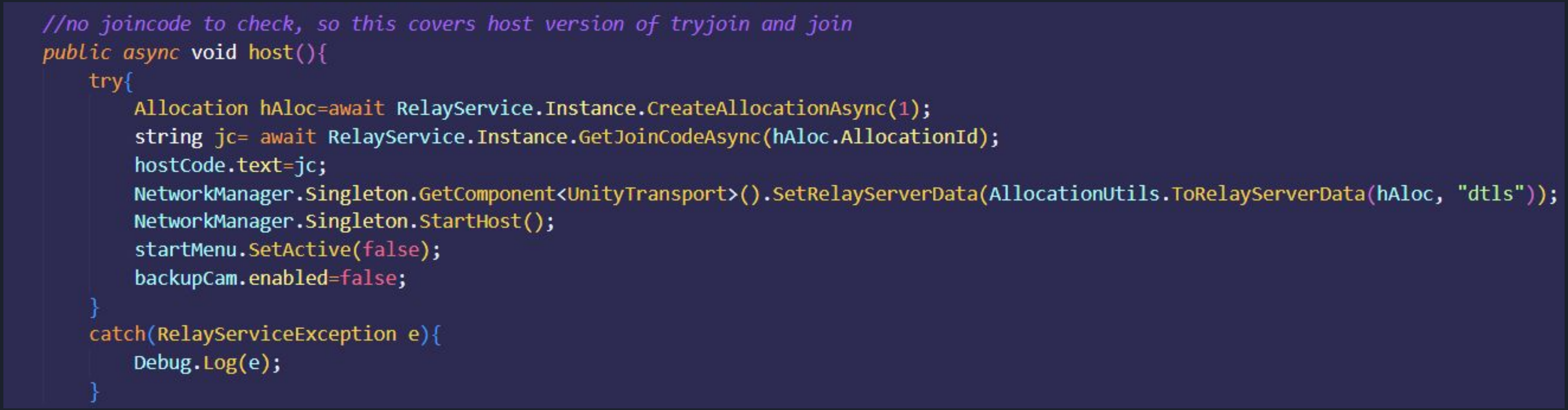
Features

- 2-player serverless online multiplayer
- Basic UI system
- Dice-based/D&D-style combat system
    - Actions/bonus actions, Attack + damage rolls, advantage/disadvantage, LOS checking
- 4 classes with basic functionalities
- 8 additional unique abilities
- World & terrain including a castle that can be walked on
- Fully animated
- Downloadable on Itch.io for free

Missing Features/Future Goals

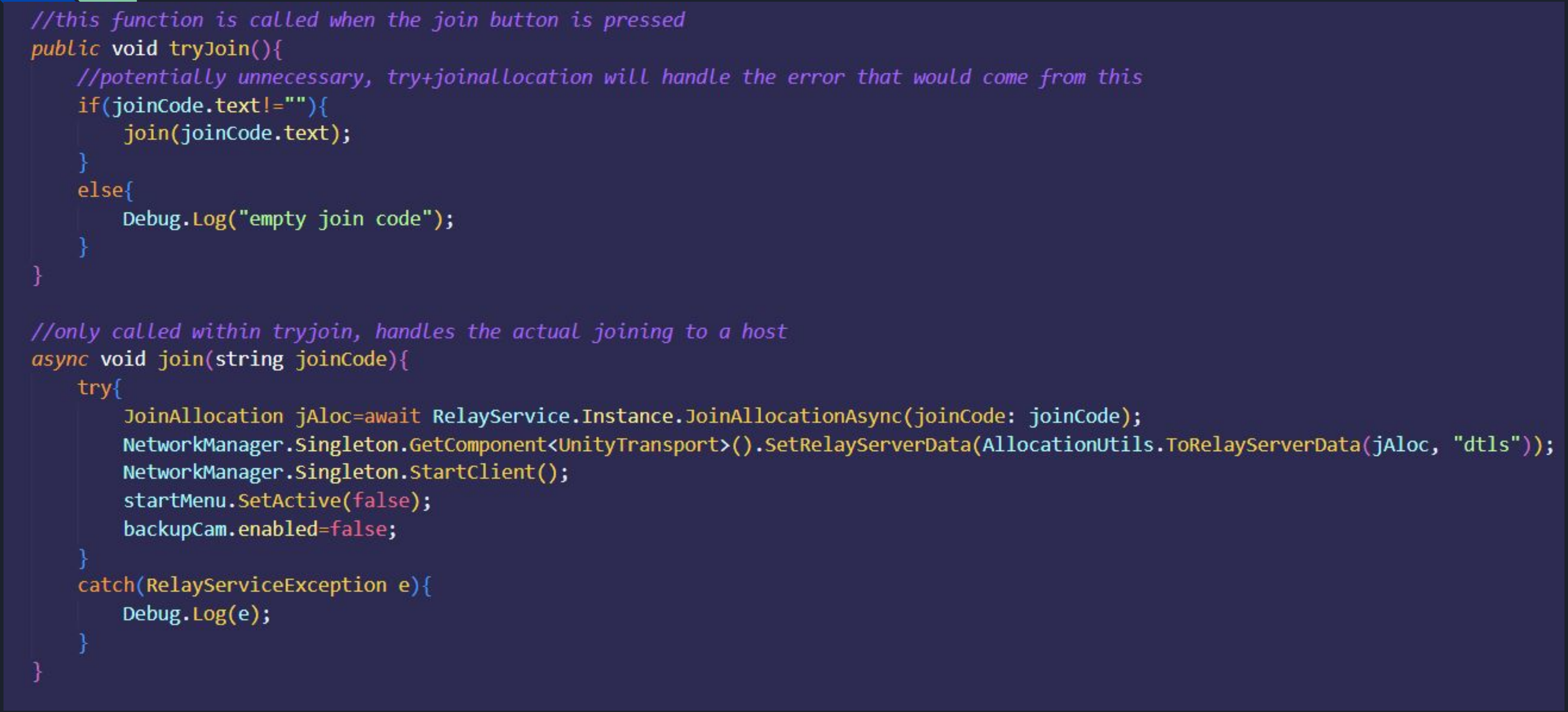
- Polished UI- ability descriptions, see actions/bonus actions remaining
- More classes/abilities
- Other D&D components- high ground, status effects, spell cost system
- Improvements to menu system- pause menu, win/loss screen, start menu
- Additions to the map- more realistic feel to the world + more locations to go to
- Ability to choose what classes are on a team
- Public lobby system- see who’s online/play against random people

Networking code- Hosting



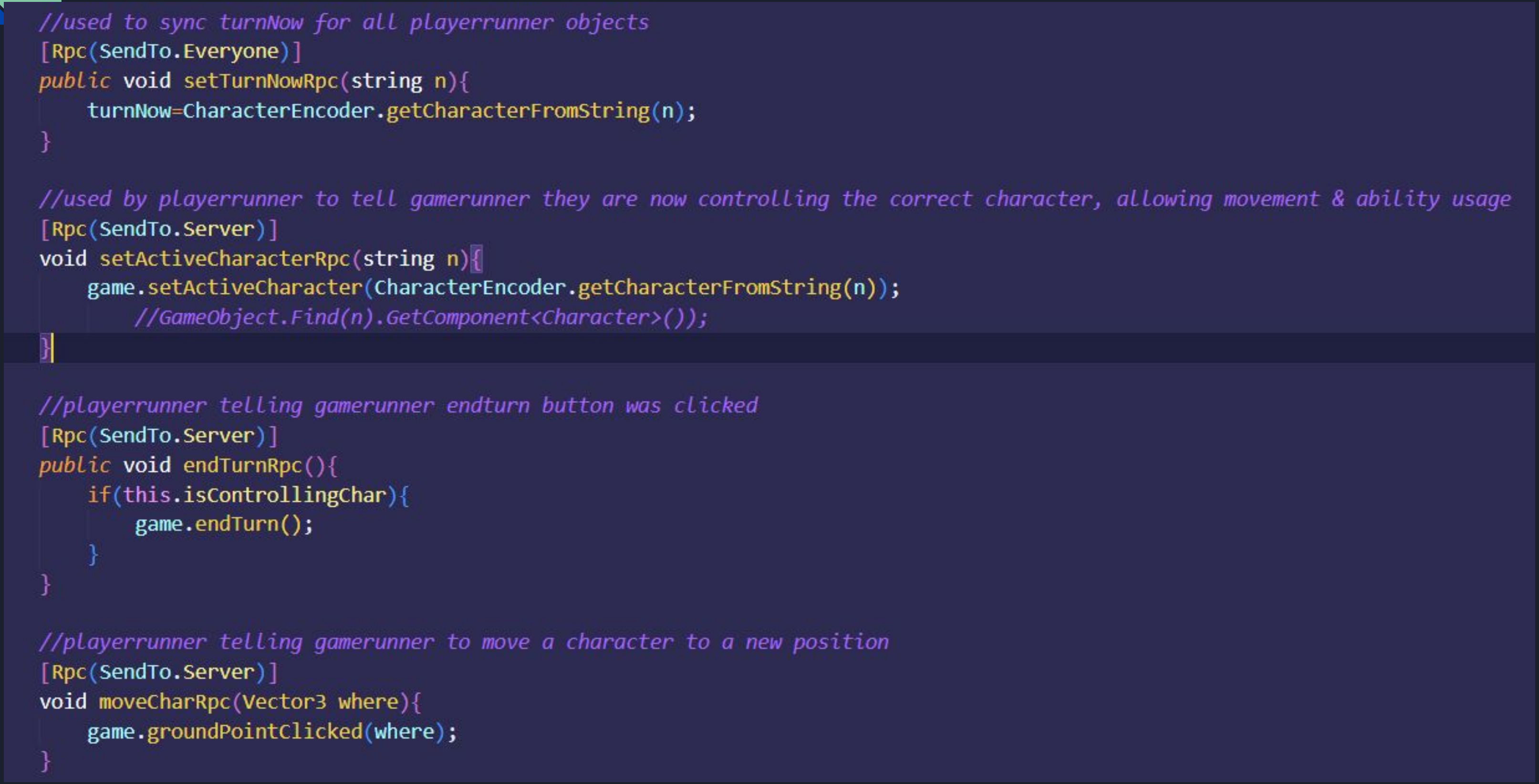
```
//no joincode to check, so this covers host version of tryjoin and join
public async void host(){
    try{
        Allocation hAloc=await RelayService.Instance.CreateAllocationAsync(1);
        string jc= await RelayService.Instance.GetJoinCodeAsync(hAloc.AllocationId);
        hostCode.text=jc;
        NetworkManager.Singleton.GetComponent<UnityTransport>().SetRelayServerData(AllocationUtils.ToRelayServerData(hAloc, "dtls"));
        NetworkManager.Singleton.StartHost();
        startMenu.SetActive(false);
        backupCam.enabled=false;
    }
    catch(RelayServiceException e){
        Debug.Log(e);
    }
```

Networking code- Joining



```
//this function is called when the join button is pressed
public void tryJoin(){
    //potentially unnecessary, try+joinallocation will handle the error that would come from this
    if(joinCode.text!=""){
        join(joinCode.text);
    }
    else{
        Debug.Log("empty join code");
    }
}

//only called within tryjoin, handles the actual joining to a host
async void join(string joinCode){
    try{
        JoinAllocation jAloc=await RelayService.Instance.JoinAllocationAsync(joinCode: joinCode);
        NetworkManager.Singleton.GetComponent<UnityTransport>().SetRelayServerData(AllocationUtils.ToRelayServerData(jAloc, "dtls"));
        NetworkManager.Singleton.StartClient();
        startMenu.SetActive(false);
        backupCam.enabled=false;
    }
    catch(RelayServiceException e){
        Debug.Log(e);
    }
}
```

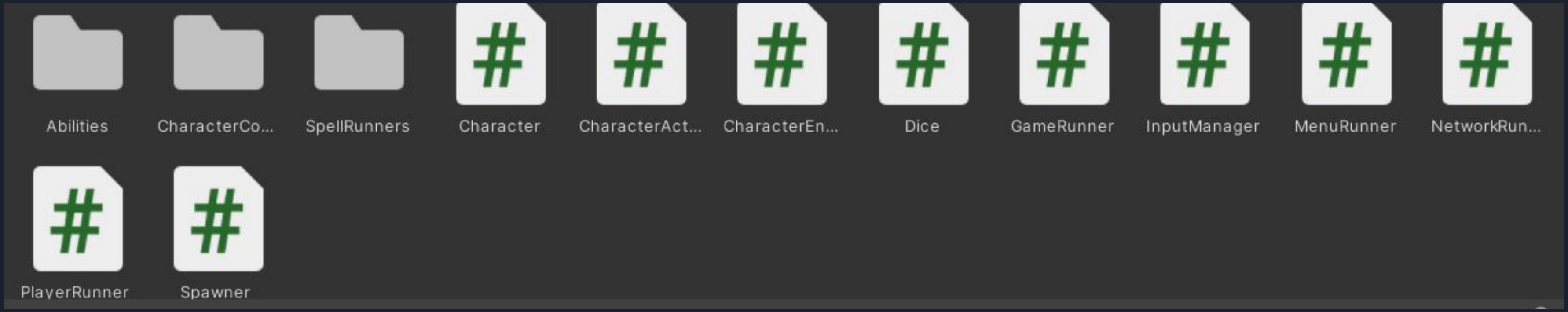
Networking code- RPCs



```
//used to sync turnNow for all playerrunner objects
[Rpc(SendTo.Everyone)]
public void setTurnNowRpc(string n){
    turnNow=CharacterEncoder.getCharacterFromString(n);
}

//used by playerrunner to tell gamerunner they are now controlling the correct character, allowing movement & ability usage
[Rpc(SendTo.Server)]
void setActiveCharacterRpc(string n){
    game.setActiveCharacter(CharacterEncoder.getCharacterFromString(n));
        //GameObject.Find(n).GetComponent<Character>());
}

//playerrunner telling gamerunner endturn button was clicked
[Rpc(SendTo.Server)]
public void endTurnRpc(){
    if(this.isControllingChar){
        game.endTurn();
    }
}

//playerrunner telling gamerunner to move a character to a new position
[Rpc(SendTo.Server)]
void moveCharRpc(Vector3 where){
    game.groundPointClicked(where);
}
```

Organization

- ~1000 lines of code
- Abilities, character controllers all inherit from base classes
- Static utility classes- Dice, CharacterEncoder



Logic Flow

Startup:

- Menu screen, NetworkRunner
- Spawner, Character initialization
- GameRunner, PlayerRunner initialization

Game Cycle:

- Camera movements handled in InputManager
- Other inputs go to PlayerRunner, sends RPCs to GameRunner
    - GameRunner may respond
- GameRunner tells Character(s) to move, trigger abilities
- Character responds with result, GameRunner relays that to PlayerRunner(s) if needed

Problems & Solutions

- Understanding how Unity/Netcode divides files into server/client-side architecture
  - Single-player game with synchronized states
- Connecting two clients together with Relay
  - Understanding events vs. override methods in C#
- Syncing data & animations between clients
  - Remote procedure calls
  - Network variables
- Handling many unique abilities in a concise manner
  - Inheritance and overriding methods

Skills Learned/Improved Upon

- Networking
  - Client/Server/Host architecture
  - Remote procedure calls
  - Netcode
  - Relay service
- Unity- NetworkBehaviour vs MonoBehaviour
- C#- subscribing to events, coroutines, inheritance
- Publishing & updating games on Itch.io

Questions?