Vedlegg D: Dokumentasjon av kode

1 CONTENTS

2 MENYSYSTEM

2.1 MENUMANAGER

MenuManager-klassen er ansvarlig for å håndtere ulike menyer og brukergrensesnitt-elementer i spillet. Den håndterer åpning og lukking av menyer, samt visning av ulike sider i spillet. Klassen har funksjoner for å åpne og lukke spesifikke menyer og sider, og den kan vise varselmeldinger. MenuManager sikrer også at riktige hendelser blir utløst når en meny eller side åpnes.

Metode	Beskrivelse
`void Awake()`	Dette er en Unity-spesifikk metode som kjører når scriptinstansen blir lastet. Den setter singleton-instansen av `MenuManager` og deaktiverer visse UI-elementer.
`void Start()`	En annen Unity-spesifikk metode, den initialiserer visse aspekter av `MenuManager`, inkludert relay managerens `OnRelayCreated`-hendelse og siden som skal vises ved starten av spillet.
`void OpenPage(MenuEnums pageToOpen)`	Denne metoden åpner den angitte siden eller menyen, avhengig av `MenuEnums`-verdien som er gitt. Den setter også synligheten til headeren og chat-visual, og aktiverer den aktuelle menyen basert på `MenuEnums` verdien. Den utløser også forskjellige hendelser basert på hvilken meny som åpnes.
`void OpenAlert(string message)`	Denne metoden brukes til å vise en varsling til spiller med en spesifikk melding.
`void CloseAlert()`	Denne metoden lukker den nåværende viste varslingen for spilleren.
`void OpenLobbyRoom(object sender, System.EventArgs e)`	Denne metoden åpner lobbyromsiden. Det er en hendelseshåndterer som reagerer når `OnRelayCreated`-hendelsen

Table 1 MenuManager

2.2 MENULAYOUTMANAGER

MenuLayoutManager-klassen er ansvarlig for å håndtere oppsettet av ulike menyer i spillet, inkludert å sette størrelser og posisjoner for UI-elementer. Klassen har funksjoner for å initialisere menyene ved å sette opp skjermstørrelsesvariabler og størrelser for UI-elementer.

Metode	Beskrivelse
`InitializeMenus()`	Initialiserer menyene ved å sette skjermstørrelsesvariabler og brukergrensesnittstørrelser.
`SetScreenSizeVariables()`	Setter skjermstørrelsesvariabler som bredde, høyde og forhold.
`SetMenuSizes()`	Setter størrelsene på menysidene.

`SetChildrenElementSizes(`
`GameObject page,`
`float buttonWidth,`
`float buttonHeight,`
`float buttonPanelHeight,`
`int MAX_VERT_ELEMENTS)`
	Setter størrelsene på barnets UI-elementer på en side.

`SetLobbyRoomChildren`
`ElementSizes(`
`GameObject page,`
`float buttonHeight,`
`float buttonPanelHeight,`
`int maxVertElements)`
	Setter størrelsene på elementene i lobbyrommet for en gitt side, ved hjelp av spesifisert knapphøyde, knapp panelhøyde, og maksimalt antall vertikale elementer.

`SetGeneralChildren`
`ElementSizes(`
`GameObject page,`
`float width,`
`float height)`
	Setter størrelsene på barnets UI-elementer på en side.

`SetChatSize()`	Setter størrelsen på chatboksen UI-element.

`SetHeaderSizeAndPosition(`
`float widtRatio,`
`float heightRatio,`
`float yOffsetRatio,`
`TextMeshProUGUI header)`
	Setter størrelse og posisjon på et overskrifts-UI-element.

Table 2 MenuLayoutManager

3 LOBBYSYSTEM

3.1 LOBBYMANAGER

LobbyManager håndterer forskjellige aspekter av en multiplayer lobby, inkludert opprettelse, innmelding, oppdatering av lobbytilstanden og spillerhåndtering innen lobbyen. Den inneholder også funksjonalitet for å opprettholde lobbyens aktivitet ved å sende regelmessige "heartbeats".

Metode	Beskrivelse
`Awake()`	Håndterer unikhet for LobbyManager-objektet ved å sørge for at det bare er en instans av det.
`Update()`	Sjekker om lobbyen er aktiv, og håndterer lobbyens polling og heartbeat hvis brukeren er vert.
`HandlePollUpdate()`	Håndterer lobbyens polling ved å regelmessig sjekke lobbyens status fra serveren.
`QueGameStart()`	Setter lobbyen klar til å starte spillet ved å oppdatere lobbyens data på serveren.
`HandleLobbyHeartBeat()`	Håndterer lobbyens heartbeat ved å sende en ping til serveren for å holde lobbyen i live.

`CreateLobby(`
`string lobbyName,`
`bool isPrivate,`
`string playerName)`
	Oppretter en ny lobby med de gitte parametrene, autoriserer relé og oppdaterer join-koden.

`UpdateJoinCode(` `string joinCode)`	Oppdaterer join-koden for lobbyen på serveren.
`QuickJoinLobby(` `string playerName)`	Lar en spiller raskt bli med i en lobby og kobles til Relay.
`ListLobbies()`	Lister opp tilgjengelige lobbyer på serveren.
`GetNewPlayer(` `string playerName)`	Returnerer en ny Player-objekt med den gitte spillerens navn.
`SetHostId(` `string hostId)`	Setter vertens id for lobbyen på serveren.
`SetPlayerReady(` `string readyConst)`	Setter en spillers ready-tilstand på serveren.
`RequestLeaveLobby()`	Lar en spiller be om å forlate lobbyen.
`StartGame()`	Setter "IsGameReady"-dataobjektet til sann på serveren.
`GetThisPlayerId()`	Returnerer id-en til gjeldende spiller.

`StopLobbyPolling()`	Denne metoden stopper pollingen for lobbyen.
`SpawnTransitionHelper()`	Denne metoden oppretter en hjelperobjekt for overgangen mellom scener.
`RegisterLobbyToGameStatusSO()`	Denne metoden registrerer lobbyinformasjonen til et ScriptableObject som representerer spilltilstanden.

Table 3 LobbyManager

3.2 LobbyRoom

LobbyRoom tar seg av den lokale delen av en lobby. Den henter regelmessige polling updates fra LobbyManager og sørger for at endringer i lobby blir reflektert visuelt for spillerne.

Metode	Beskrivelse
`Awake()`	Starter metoder for å håndtere knapper ved initialisering av objektet.
`Start()`	Abonnerer på hendelsen OnLobbyRoomOpened fra MenuManager ved oppstart av objektet.
`MenuManager_ OnLobbyRoomOpened( object sender, EventArgs e)`	Initialiserer spillere, setter headeren, og abonnerer på visse hendelser når en lobby er åpnet.
`InitPlayers()`	Initialiserer spillerne ved å fylle listene med spiller navn og spiller klare tekst, samt deaktiverer alle spiller og klare objekter.
`Subscribe()`	Kobler LobbyRoom til MenuManager og LobbyManager ved å abonnere på forskjellige hendelser.
`HandleReadyButton()`	Håndterer funksjonaliteten til "klar"-knappen, hvor spilleren kan markere seg selv som klar eller ikke klar.
`HandleStartGameButton()`	Gjør StartGameButton ikke-interaktiv og legger til en lytter for å kø oppstart av spillet.
`HandleLeaveButton()`	Legger til en lytter for å anmode om å forlate lobbyen.
`SetHeader()`	Setter lobby navn og kode tekst.
`HandlePollUpdate( object sender, EventArgs lobbyEventArg)`	Håndterer lobby oppdateringer basert på lobby polling event.
`UpdateLocalLobby( Lobby lobby)`	Oppdaterer lokale lobby data basert på spiller informasjon fra lobbyen.
`LoadNetwork(bool isHost)`	Forbereder NetworkManager for spillstart basert på vertsstatus.
`NetworkManager_ ConnectionApproval Callback( NetworkManager .ConnectionApprovalRequest connectionApprovalRequest, NetworkManager .ConnectionApprovalResponse connectionApprovalResponse)`	Godkjenner tilkoblingsforespørsler.

`OnLobbyLeft( object sender, EventArgs e)`	Håndterer lobby forlate hendelsen, åpner lobby menyen når det skjer.

Table 4 LobbyRoom

4 CHAT-SYSTEM

4.1 CHATMANAGER

ChatManager tar seg av chat-meldinger mellom klienter.

Metode	Beskrivelse
`ChatManager()`	En konstruktør som initialiserer en ny instans av `ChatManager` klassen.
`SetMaxMessages(int max)`	Setter maksimalt antall meldinger som kan vises i chat-systemet.
`Awake()`	Denne metoden blir kalt når script-instansen blir lastet. Den sørger for at det bare er en aktiv instans av `ChatManager` i spillet.
`Start()`	Denne metoden blir kalt før den første rammeoppdateringen. Den legger til en listener på chat-input-feltet, og setter en karaktergrense.
`Update()`	Denne metoden blir kalt ved hver rammeoppdatering. Den bygger chat-innholdet regelmessig og oppdaterer fokus-statusen til chat-input-feltet.
`SubmitMsg()`	Sender den innskrevne meldingen fra spilleren, og aktiverer deretter input-feltet igjen og tømmer det.
`AddMsg(string message, ulong senderPlayerId, string senderName)`	Legger til en melding i meldingslisten fra en bestemt spiller. Hvis meldingslisten blir for lang, fjernes den eldste meldingen.
`SendMsg(string message, string senderName)`	Sender en melding fra en spiller, og kaller på serverfunksjonen for å distribuere meldingen.
`SendChatMessageServerRpc(string message, string senderName, ServerRpcParams serverRpcParams)`	Server RPC for å sende en chat-melding.
`ReceiveChatMessageClientRpc(string message, ulong senderPlayerId, string senderName)`	Klient RPC for å motta en chat-melding.
`BuildChatContents()`	Bygger chat-innholdet fra meldingslisten.

Table 5 ChatManager

5 RELAY

5.1 RELAYMANAGER

RelayManager har som oppgave å koble spillet opp mot Unity sin Relay tjeneste.

Metode	Beskrivelse
`RelayManager()`	Konstruktør som initialiserer en ny instans av RelayManager klassen.
`Awake()`	Denne metoden blir kalt når script-instansen blir lastet. Den sørger for at det bare er en aktiv instans av RelayManager i spillet.
`Authorize()`	Initialiserer Unity-tjenester og logger inn anonymt.
`InitAsyncWithProfile()`	Initialiserer Unity-tjenester med en profil. Prøver flere ganger hvis initialiseringen mislykkes.
`CreateRelay()`	Oppretter en relay-allokasjon og starter verten. Returnerer en Dictionary med relay join-kode og allokasjons-ID.
`JoinRelay(` `string joinCode)`	Bli med i en relay ved hjelp av den gitte join-koden.
`JoinRelayShortcut(` `string joinCode)`	Bli med i en relay ved hjelp av den gitte join-koden (snarveiversjon). Koden skal bare ha 6 tegn.

Table 6 RelayManager

6 OVERGANG MELLOM SCENER

6.1 TRANSITIONHELPER

TransitionHelper befinner seg på en prefab som blir spawnet. Den har dermed et netcode komponent knyttet til seg som gjør at den kan sende RPC meldinger. Dette bruker den til å kommunisere til server når spillere er klar til å starte spillet.

Metode	Beskrivelse
`OnNetworkSpawn()`	Metoden blir kalt når et nettverksobjekt blir spawnet. Den registrerer lobbyen til GameStatusSO og kaller på AddPlayerReadyServerRpc().

`OnPlayerReadyCountChanged(`
`int previousValue,`
`int newValue)`
	Metoden laster inn spillets scene når alle spillerne er klare.

`AddPlayerReadyServerRpc()`	Denne ServerRpc registrerer en ny spiller som klar ved å øke playerReadyCount med 1.

Table 7 TransitionHelper

6.2 LOADINGMANAGER

LoadingManager tar for seg loading-skjermen når spillet går fra meny-scenen til game-scenen. Den bestemmer hva som skal vises og skjules samt genererer en nedtelling for spiller før spillet starter. Når nedtellingen er over deaktiverer den loading-skjermen så spillet blir synlig.

Metode	Beskrivelse
`Awake()`	Aktiverer loading screen, deaktiverer powerupUI og instansierer touchUI hvis spillet kjøres på Android.
`ServerStart()`	Starter server-spesifikke atferd, inkludert å vente på at spillerne er klare og starte GameManager.
`Start()`	Starter forskjellige atferd avhengig av om klienten er en server, setter opp event-handlere og deaktiverer chatPanelet.

`OnCountdownFinished(`
`bool previousValue,`
`bool newValue)`
	Utføres når nedtellingen er ferdig. Deaktiverer lasteskjermen, aktiverer powerupUI og chatPanel, og gjør spillerkontrollene aktive. Aktiverer touch-kontroller hvis spillet kjøres på Android.

`OnClientsReady(`
`bool prev,`
`bool newValue)`
	Starter nedtellingen når alle klientene er klare.

`WaitForPlayersReady()`	Venter på at alle spillere skal være klare før den fortsetter. Sjekker om antallet tilkoblede klienter samsvarer med antall lobby-spillere.
`StartCountdown()`	Starter pre-game nedtellingen. Nedtellingens varighet er 3 sekunder.

Table 8 LoadingManager

7 KARTSYSTEM

7.1 TILEMAPMANAGER

Ønsker å endre. Venter med denne.

8 SENTRALISERT SPILL-LOGIKK

8.1 LOCALPLAYERMANAGER

LocalPlayerManager har som oppgave å lagre informasjon om lokal spiller.

Metode	Beskrivelse
Awake()	Initialiserer localPlayer som levende, setter Singleton-instansen, og deaktiverer ulike spillobjekter (powerups og ring).
IsThisIdForLocalPlayer(ulong id)	Sjekker om den gitte spiller-ID-en matcher ID-en til den lokale spilleren.
GetNameFromId(ulong id)	Returnerer navnet til spilleren hvis den gitte ID-en matcher ID-en til den lokale spilleren, ellers returnerer en tom streng.
SetName(string name)	Setter navnet til den lokale spilleren.
SetId(ulong id)	Setter ID-en til den lokale spilleren.
SetHasRing(bool playerHasRing)	Setter 'HasRing'-statusen til den lokale spilleren.
SetPlayerWonGame(bool playerWonGame)	Setter 'PlayerWonGame'-statusen til den lokale spilleren.
RegisterPlayerInScriptableObject()	Registrerer den lokale spilleren i gameStatusSO scriptable objektet ved hjelp av data fra AuthenticationService.

Table 9 LocalPlayerManager

8.2 GAMEMANAGER

GameManager har sentralisert spill-logikk.

Metode	Beskrivelse
Awake()	Initialiserer GameManager singleton-instansen og deaktiverer largeMessage-spillobjektet.
Start()	Registrerer den lokale spilleren i ScriptableObject ved hjelp av LocalPlayerManager.
StartGameManagerServer()	Starter GameManager på serveren og styrer spawing av objekter.
EndGameScene()	Håndterer avslutning av spillscenen og returnerer til hovedmenyen, fjerner også alle "DontDestroy"-taggede objekter.
OnPlayerDeath()	Håndterer hendelsen når en spiller dør.

DestroyObjects TaggedWithDontDestroy()	Fjerner objekter merket med "DontDestroy" når man returnerer til meny-scenen.
OnPlayerPickedUpRing(string playerName)	Håndterer hendelsen når en spiller plukker opp en ring.
VisualizeOnGameWon(string playerName)	Håndterer visualiseringen når en spiller vinner spillet.
DeactivateAllPlayers()	Deaktiverer alle spillerne i spillet.

Table 10 GameManager

9 SPILLER-LOGIKK

9.1 PLAYERBEHAVIOUR

Metode	Beskrivelse
Awake(): void	Initialiserer visualisering for touch controller på Android
Start(): void	Abonnerer på hendelse som avgjør om chat er i fokus eller ikke
Initialize(): void	Abonnerer på helsestatus til avatar. Initialiserer animator.
OnPlayerKnockdown(object sender, PlayerHealth.OnPlayerKnockdownEventArgs e): void	Gir beskjed om at en spiller er slått ut.
OnNetworkSpawn() : void	Initialiserer komponenter for karakterkontroll og spiller input mth. mobil. Initialiserer lys-effekt. Setter posisjon til avatar. Kaller Initialise()
LateUpdate() : void	Flytter kameraet i forhold til avatarens posisjon
Update() : void	Sjekker om avatar er slått ut, om chat er i fokus og velger kontroller i henhold til plattform
HandleMovement() : void	Håndterer spiller input og forflytning av avataren på Windows
HandleTouchInput(): void	Håndterer spiller input forflytning av avataren på Android
Attack(): void	Angrep
SetNewSword() : void	Oppgraderer Sverdangrepet

GetSword() : bool	Avgjør om avataren har tresverd eller stålsverd
IncreaseSpeed(): void	Øker farten i forbindelse med Power Up
ResetSpeedAfterDelay(float delay) : IEnumerator	Resetter avatarens fart etter å ha brukt opp Power Up
RelocatePlayer(Vector2 cavePosition): void	Transporterer avataren til et nytt kart
ActivateTorchPowerUp(): void	Avgjør om vi skal tenne en ny fakkel eller ikke
FireUpNewTorch(float delay) : IEnumerator	Tenner opp en ny fakkel i et angitt tidsintervall

10 NPC-LOGIKK

10.1 ENEMY

Metode	Beskrivelse
Awake(): void	Fienden starter å vandre omkring
Start(): void	Setter tilfeldig tidsrom for hvor lenge det skal gå før fienden tar en avgjørelse på hvilken retning den skal ta
Update(): void	Begrenser angrep til en gang pr. sekund
OnPlayerKnockdown(object sender, PlayerHealth.OnPlayerKnockdownEventArgs e): void	Gir beskjed om hva fienden skal gjøre når spilleren er nede. Fjerner en av spillerne. Stopper animasjon. Endrer status til State.PlayerDown
OnNetworkSpawn() : void	Initialiserer komponenter for karakterkontroll og spiller input mth. mobil. Initialiserer lys-effekt. Setter posisjon til avatar. Kaller Initialise()
ChooseMoveDirection(): void	Velger en tilfeldig retning nord, sør, øst eller vest
OnNetworkSpawn(): void	Cacher spillerne i en liste
FixedUpdate(): void	Inneholder de forskjellige statusene fienden kan befinne seg i, State.Roaming, State.PlayerDown eller State.ChasingTarget
BreakBeforeRoaming(): IEnumerator	Fienden tar en pause før den vandrer videre etter at avataren den jaget er nedkjempet

SearchForTarget(): void	Søker etter avatarer innenfor forhåndsbestemt radius
ChaseTarget(): void	Forfølger avataren hvis den ikke kommer utenfor en forhåndsbestemt distanse. Angriper hvis den kommer nær nok avataren
OnTriggerEnter2D: void	Skifter target hvis en annen avatar kommer i kontakt med en Circle Collider 2D som omgir fienden
StopAnimationClientRpc(): void	Server sender beskjed til alle klientene om å stoppe animasjonen til fienden etter at den har nedkjempet avataren den jaktet på.
Attack()	Fienden angriper med stålsverd