SE 3XA3: Module Interface Specification
Zombie Survival Kit

Team #6, Group 6ix
Mohammad Hussain hussam17
Brian Jonatan jonatans
Shivaansh Prasann prasanns

December 1, 2018

Modules

M1: Item (Component)

M2: ConsumableItem (Component)

M3: EquipmentItem (Component)

M4: EquipmentManager (Manager)

M5: InventoryManager (Manager)

M6: Interactable (Object)

M7: Enemy (Object)

M8: ItemStore (Object)

M9: InteractableController (Character)

M10: EquipmentUI

M11: EquipmentSlotUI

M12: InventoryUI

M13: InventorySlotUI

M14: CharacterCombat (Character)

M15: CharacterStats (Manager)

M16: PlayerStats (Manager)

M17: ZombieStats (Manager)

M18: Stat (Component)

M19: Zombie (Object)

M20: FirstPersonController (Character)

M21: Gun (Object)

M22: BulletDamage (Objects)

M23: DayLightController

M24: PlayerUI

M1 Item

Template Module

Item

Uses

M5

Syntax

Exported Types

Sprite

Exported Access Programs

Routine name	In	Out	Exceptions
Use			
RemoveFromInventory			

Semantics

State Variables

name: *String*
icon: *Sprite*

State Invariant

None

Assumptions

This module is used to create a new asset in Unity by creating a new asset menu called "Inventory/Item". Once an item has been created through Unity's asset menu, the state variables are updated directly in Unity by manually typing in the name of the item, and placing the appropriate Sprite for the icon. User will not be tasked to do this; all items available to the user will be created before hand by the developers of Zombie Survival Kit.

Access Routine Semantics

Use():

- translation: None
- output: Prints out to Debug.Log()
- exception: None

RemoveFromInventory():

- translation: Calls the "Remove" method from the Inventory class.
- output: None
- exception: None

M2 ConsumableItem

Template Module

ConsumableItem

Uses

M1, M5

Syntax

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Use			

Semantics

State Variables

healthModifier: $\mathbb{N}$
name: *String*
icon: *Sprite*

State Invariant

None

Assumptions

This module is used to create a new asset in Unity by creating a new asset menu called "Inventory/Consumable"; item of type Consumable. Once a Consumable has been created through Unity's asset menu, the state variables are updated directly in Unity by manually typing the value of the healthModifier, the name of the item, and placing the appropriate Sprite for the icon. User will not be tasked to do this; all Consumable items available to the user will be created before hand by the developers of Zombie Survival Kit.

Access Routine Semantics

Use():

- translation: Calls the base "Use" method from the Item class, the "Eat" method in the PlayerStats class, and "RemoveFromInventory" method from the Item class.
- output: None
- exception: None

M3 EquipmentItem

Template Module

EquipmentItem

Uses

Item, EquipmentManager, System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Types

equipmentSlot: {Head, Chest, Legs, Primaryhand, Offhand, Feet}

Exported Access Programs

Routine name	In	Out	Exceptions
Use			

Semantics

State Variables

equipSlot: *equipmentSlot*
attackModifier: $\mathbb{N}$
defenceModifier: $\mathbb{N}$
name: *String*
icon: *Sprite*

State Invariant

None

Assumptions

This module is used to create a new asset in Unity by creating a new asset menu called "Inventory/Equipment"; item of type EquipmentItem. Once an EquipmentItem has been created through Unity's asset menu, the state variables are updated directly in Unity by manually choosing which equipmentSlot belongs to the EquipmentItem's equipSlot, typing in the value of the attackModifier, defenceModifer and the name of the EquipmentItem,

and placing the appropriate Sprite for the icon. User will not be tasked to do this; all EquipmentItem available to the user will be created before hand by the developers of Zombie Survival Kit.

Access Routine Semantics

Use():

- transition: Calls the base "Use" method from the Item class, the "Equip" method in the EquipmentManager class, and "RemoveFromInventory" method from the Item class.
- output: None
- exception: None

M4 EquipmentManager

Template Module

EquipmentManager

Uses

M1, M4, M11

Syntax

Exported Types

EquipmentItem, Inventory, equipmentSlot

Exported Access Programs

Routine name	In	Out	Exceptions
OnEquipmentChanged	EquipmentItem, EquipmentItem		
Start			
Update			
Equip	EquipmentItem		
Unequip	$\mathbb{N}$		
UnequipAll			
IsGunEquipped		$\mathbb{B}$	
IsAxeEquipped		$\mathbb{B}$	

Semantics

State Variables

equippedItems: array of *EquipmentItem*
inventory: *InventoryManager*
axeUI: *GameObject*
gunUI: *GameObject*
fpsController: *GameObject*
gun: *GameObject*
equippedGun: *GameObject*
isGunEquipped: $\mathbb{B}$

isAxeEquipped: $\mathbb{B}$

State Invariant

instance: *EquipmentManager*

Assumptions

This module updates automatically as the user changes the player's equipment by equipping or unequipping EquipmentItems

Access Routine Semantics

OnEquipmentChanged(EquipmentItem newEquipment, EquipmentItem oldEquipment):

- description: Is called whenever a change occurs in the the state variable equippedItems. It is a delegate function that allows itself to perform different functionalities in different modules.*
- translation: *Updated in different modules; does nothing in this module*
- output: None
- exception: None

Start():

- translation: Initializes state variable equippedItems with the length of equipmentSlot (an enum type), state variable inventory as an instance of the Inventory module, and the fpsController/gunUI/axeUI objects by finding game objects with their respective tags.
- output: None
- exception: None

Update():

- translation: Upon keyboard input of "U", perform UnequipAll(). Also, disable both the axe and gun UI if there is no item in the primary hand
- output: None

- exception: None

Equip(EquipmentItem newEquipment):

- translation: Stores the newEquipment item into equippedItems[newEquipment.equipSlot]. If the newEquipment is a gun: update the isGunEquipped variable, enable the gunUI, and set equippedGun to instantiate a gun object at the location of the fpsController. If the newEquipment is an axe: update the isAxeEquipped variable, and enable the axeUI.
- output: None
- exception: None

Unequip(int SlotIndex):

- translation: Removes the EquipmentItem in equippedItems[SlotIndex]. If the old item is a gun: update the isGunEquipped variable, disable the gunUI, and set destroy the created gun object. If the old item is an axe: update the isAxeEquipped variable, and disable the axeUI.
- output: None
- exception: None

UnequipAll():

- translation: Removes all EquipmentItem in equippedItems.
- output: None
- exception: None

IsGunEquipped():

- translation: A public method to return the private variable isGunEquipped.
- output: isGunEquipped
- exception: None

IsAxeEquipped():

- translation: A public method to return the private variable isAxeEquipped.
- output: isAxeEquipped
- exception: None

M5 InventoryManager

Template Module

Inventory

Uses

M1

Syntax

Exported Types

Item, List

Exported Access Programs

Routine name	In	Out	Exceptions
OnItemChanged			
Add	Item	bool	
Remove	Item		

Semantics

State Variables

items: $List < Item >$

State Invariant

instance: *Inventory* space: $\mathbb{N}$

Assumptions

This module updates automatically as the user changes the player's inventory by interacting with GameObjects that are available to be stored in the inventory.

Access Routine Semantics

OnItemChanged():

- description: Is called whenever a change occurs in the the state variable items. It is a delegate function that allows itself to perform different functionalities in different modules.
- translation: *Updated in different modules; does nothing in this module*
- output: None
- exception: None

Add(Item item):

- translation: Adds item to the state variable list items if the number of Item in the list is less than state invariant space.
- output: Returns true if an item is added to the list; returns false if an item is not added to the list
- exception: None

Remove(Item item):

- translation: Removes item from the state variable list item.
- output: None
- exception: None

M6 Interactable

Template Module

Interactable

Uses

None

Syntax

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Interact Update onFocused onDefocused onDrawGizmosSelected	Transform		

Semantics

State Variables

isFocus: *bool*
player: $Transform$
hasInteracted: *bool*

State Invariant

radius: $\mathbb{Q}$

Assumptions

This module is attatched to any GameObject that the player can interact with.

Access Routine Semantics

Interact():

- description: It is a Virtual function that is re-programmable in all other modules that inherit M6.
- translation: *Updated in different modules; does nothing in this module*
- output: None
- exception: None

Update():

- translation: If isFocus = true, and hasInteracted = false, and if the distance between the player and the interactable GameObject is less than state invariant radius, perform Interact() and set hasInteracted = true.
- output: None
- exception: None

onFocused(Transform playerTransform):

- translation: sets isFocus = true, player = playerTransform, and hasInteracted = true.
- output: None
- exception: None

onDefocused():

- translation: sets isFocus = false, player = null, and hasInteracted = false.
- output: None
- exception: None

onDrawGizmosSelected():

- translation: Draws a yellow wire sphere around the interactable object of radius state invariant radius that is only visible in the scene view of Unity.
- output: None
- exception: None

M7 Enemy

Template Module

Enemy

Uses

M1, M14, M15, M16

Syntax

Exported Types

PlayerStats, CharacterStats, CharacterCombat

Exported Access Programs

Routine name	In	Out	Exceptions
Start Interact			

Semantics

State Variables

playerManager: *PlayerStats*
enemyStats: *CharacterStats*

State Invariant

None

Assumptions

This module inherits from the Interactable class.

Access Routine Semantics

Start():

- translation: Initializes enemyStats with the current CharacterStats component on the zombie game object that this script is attached to and playerManager with an instance of the player so the program knows where the attack is coming from.
- output: None
- exception: None

Interact():

- translation: Calls the base Interact method from Interactable and creates a CharacterCombat (CC) variable with the CC component from the playerManager. Calls the Attack() method on this CC variable with the enemyStats variable as the parameter for whose stats to affect as a result of the player attack.
- output: None
- exception: None

M8 ItemStore

Template Module

ItemStore

Uses

M1, M6, M5

Syntax

Exported Types

Item, Inventory

Exported Access Programs

Routine name	In	Out	Exceptions
Interact StoreItem			

Semantics

State Variables

item: *Item*

State Invariant

None

Assumptions

This module is attatched to any GameObject that the player can interact with.

Access Routine Semantics

Interact():

- description: It is an override function that is re-programs Interact() (from M6) to perform StoreItem().
- translation: *Updated in different modules; does nothing in this module*
- output: None
- exception: None

StoreItem():

- translation: Stores the state variable item into the inventory and destroys the GameObject.
- output: None
- exception: None

M9 InteractableController

Template Module

InteractableController

Uses

M6

Syntax

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
pickup			
AttackEnemy			
SetFocus	Interactable		
RemoveFocus			

Semantics

State Variables

focus: *Interactable*
hit: *RaycastHit*
player: *GameObject*
smooth: $\mathbb{Q}$

State Invariant

distanceToSee = $\mathbb{Q}$

Assumptions

This module is attatched to the GameObject with tag "MainCamera".

Access Routine Semantics

Start():

- translation: Initializes player with the GameObject with the tag "MainCamera"
- output: None
- exception: None

Update():

- translation: Performs pickup() and AttackEnemy()
- output: None
- exception: None

pickup():

- translation: If the the user performs an keyboard input of "E", and if the Raycast collides with an Interactable GameObject, perform SetFocus() with the interactable component attatched to the GameObject.
- output: None
- exception: None

AttackEnemy():

- translation: If the the user performs a left-mouse button input, and if the Raycast collides with an Interactable GameObject, perform SetFocus() with the interactable component attatched to the GameObject.
- output: None
- exception: None

SetFocus(Interactable newFocus):

- translation: If newFocus $\neq$ focus (state variable), calls the Interactable method onDefocused() on focus, and if focus $\neq$ null and set's focus = newFocus. Calls the Interactable method onFocused on the transform of the newFocus.
- output: None

- exception: None

RemoveFocus(Interactable newFocus):

- translation: If focus (state variable) $\neq$ null, calls the Interactable method onDefocused() on focus. Set's focus = null.
- output: None
- exception: None

M10 EquipmentUI

Template Module

EquipmentUI

Uses

M4, M11

Syntax

Exported Types

Transform, Canvas, EquipmentManager, EquipmentSlot,

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
UpdateEquipmentUI	EquipmentItem, EquipmentItem		

Semantics

State Variables

equipmentParent: $Transform$
equipmentUI: $Canvas$

State Invariant

equipment: $EquipmentManager$
equipmentSlots: array of $EquipmentSlot$

Assumptions

All the state variables are initialized by manually dragging GameObjects into the appropriate slot in Unity.

Access Routine Semantics

Start():

- translation: Initializes State Invariant equipment as an instance of EquipmentManager, and equipmentSlots is initialized to store all the GameObjects with the component EquipmentSlot.
  The EquipmentManager method onEquipmentChanged() is re-programmed to perform the method UpdateEquipmentUI().
- output: None
- exception: None

Update():

- translation: Upon keyboard input of "I", check if the equipmentUI is enabled (showing on screen). If it is enabled, de-enable it, else, enable it.
- output: None
- exception: None

UpdateEquipmentUI(EquipmentItem New, EquipmentItem Old):

- translation: If New = null, perform EquipmentSlot method clearSlot on equipmentSlots[(int)Old.equipSlot]. Else, perform EquipmentSlot method addItem() on equipmentSlots[(int)New.equipSlot] using New as the paramater for addItem().
- exception: None

M11 EquipmentSlotUI

Template Module

EquipmentSlot

Uses

M3, M4

Syntax

Exported Types

Image, Button, EquipmentItem

Exported Access Programs

Routine name	In	Out	Exceptions
addItem clearSlot onRemoveButton	EquipmentItem		

Semantics

State Variables

icon: *Image* removeButton: *Button* item: *EquipmentItem* slotNumber: $\mathbb{N}$

State Invariant

None

Assumptions

A module used as a component to GameObjects. State variable slotNumber will have it's value manually inputted.

Access Routine Semantics

addItem(EquipmentItem newEquipment):

- translation: Set's state variable item = newEquipment, set's the sprite attribute of icon to the icon attribute of item (icon.sprite = item.icon), sets the enabled attribute of icon to true (icon.enabled = true), and sets the interactable attribute of removeButton to true (removeButton.interactable = true).
- output: None
- exception: None

clearSlot():

- translation: Set's state variable item = null, set's the sprite attribute of icon to null (icon.sprite = null), sets the enabled attribute of icon to false (icon.enabled = false), and sets the interactable attribute of removeButton to false (removeButton.interactable = false).
- output: None
- exception: None

onRemoveButton():

- translation: Calls the EquipmentManager method Unequip() using state variable slotNumber as the parameter.
- output: None
- exception: None

M12 InventoryUI

Template Module

EquipmentUI

Uses

M5, M8

Syntax

Exported Types

Transform, Canvas, Inventory, InventorySlot,

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
UpdateInventoryUI	EquipmentItem, EquipmentItem		

Semantics

State Variables

itemsParent: $Transform$
inventoryUI: $Canvas$

State Invariant

inventory: $Inventory$
slots: array of $InventorySlot$

Assumptions

All the state variables are initialized by manually dragging GameObjects into the appropriate slot in Unity.

Access Routine Semantics

Start():

- translation: Initializes State Invariant inventory as an instance of Inventory, and slots is initialized to store all the GameObjects with the component InventorySlot. The Inventory method onInventoryChanged() is re-programmed to perform the method UpdateInventoryUI().
- output: None
- exception: None

Update():

- translation: Upon keyboard input of "I", check if the inventoryUI is enabled (showing on screen). If it is enabled, de-enable it, else, enable it.
- output: None
- exception: None

UpdateEquipmentUI():

- translation: The InventorySlot method addItem() is called for every item in the Inventory state variable List items, and the InventorySlot method clearSlot() is called for every element in slots that does not have an item.
- output: None
- exception: None

M13 InventorySlotUI

Template Module

EquipmentSlot

Uses

M5

Syntax

Exported Types

Image, Button, Item

Exported Access Programs

Routine name	In	Out	Exceptions
addItem clearSlot onRemoveButton useItem	Item		

Semantics

State Variables

icon: *Image* removeButton: *Button* item: *Item*

State Invariant

None

Assumptions

A module used as a component to GameObjects. State variable slotNumber will have it's value manually inputted.

Access Routine Semantics

addItem(Item newItem):

- translation: Set's state variable item = newItem, set's the sprite attribute of icon to the icon attribute of item (icon.sprite = item.icon), sets the enabled attribute of icon to true (icon.enabled = true), and sets the interactable attribute of removeButton to true (removeButton.interactable = true).
- output: None
- exception: None

clearSlot():

- translation: Set's state variable item = null, set's the sprite attribute of icon to null (icon.sprite = null), sets the enabled attribute of icon to false (icon.enabled = false), and sets the interactable attribute of removeButton to false (removeButton.interactable = false).
- output: None
- exception: None

onRemoveButton():

- translation: Calls the Inventory method Remove() using state variable item as the parameter.
- output: None
- exception: None

useItem():

- translation: If item $\neq$ null, perform the Item method Use() on state variable item.
- output: None
- exception: None

M14 CharacterCombat

Template Module

CharacterCombat

Uses

M15, M16

Syntax

Exported Types

CharacterStats, PlayerStats

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
Attack	CharacterStats		
~~DoDamage~~	~~CharacterStats, $\mathbb{R}$~~	~~IEnumerator~~	

Semantics

State Variables

attackerStats: *CharacterStats*
playerManager: *PlayerStats*
~~attackSpeed: $\mathbb{R}$~~
attackCooldown: $\mathbb{R}$
attackDelay: $\mathbb{R}$

State Invariant

~~attackDelay = 1.5~~

Assumptions

This module requires the component to be of type CharacterStats.

Access Routine Semantics

Start():

- translation: Initializes attackerStats with the CharacterStats component on the game object this script is attached to.
- output: None
- exception: None

Update():

- translation: Subtracts "Time.deltaTime" from the attackCooldown every Update.
- output: None
- exception: None

Attack(CharacterStats targetStats):

- translation: Runs the ~~DoDamage() method~~ TakeDamage() function stats parameter with the attackerStats damage value once the attackCooldown reaches 0. Resets the cooldown back to 1 divided by the attackSpeed and plays a sound effect through the playerManager.
- output: None
- exception: None

~~DoDamage(CharacterStats stats, float delay):~~

- ~~translation: Runs the TakeDamage() function on the stats parameter with the attackerStats damage value.~~
- ~~output: *out* := *WaitForSecondsDelay*(*delay*), so the attack only commences after the delay timer is over.~~
- ~~exception: None~~

M15 CharacterStats

Template Module

CharacterStats

Uses

M18

Syntax

Exported Types

Stat

Exported Access Programs

Routine name	In	Out	Exceptions
Start TakeDamage Die	$\mathbb{N}$		

Semantics

State Variables

maxHealth: $\mathbb{N}$
curHealth: $\mathbb{N}$
dmg: *Stat*
armour: *Stat*

State Invariant

maxHealth = 100

Assumptions

None

Access Routine Semantics

Start():

- translation: Initializes curHealth as the maxHealth.
- output: None
- exception: None

TakeDamage(int damage):

- translation: Subtracts the armour from the damage parameter and substracts this modified damage from the curHealth. If the health goes below 0, Die() is called.
- output: None
- exception: None

Die():

- translation: Only displays a log message as this method is overwritten in PlayerStats/ZombieStats.
- output: None
- exception: None

M16 PlayerStats

Template Module

PlayerStats

Uses

M2, M3, M4, M15, M18

Syntax

Exported Types

PlayerStats, EquipmentManager

Exported Access Programs

Routine name	In	Out	Exceptions
Awake			
Start			
OnEquipmentChanged	EquipmentItem, EquipmentItem		
Eat	ConsumableItem		
Die			
MeleeSound			

Semantics

State Variables

instance: *PlayerStats*
player: *GameObject*
equipmentManager: *EquipmentManager*
axeSound: *AudioClip*
punchSound: *AudioClip*

State Invariant

None

Assumptions

This module inherits from CharacterStats.

Access Routine Semantics

Awake():

- translation: Initializes the PlayerStats instance to "this" instance of the class (for a constant reference to the Player Object in Unity)
- output: None
- exception: None

Start():

- translation: Initializes curHealth as maxHealth, the equipmentManager, and the onEquipmentChanged delegate inside the EquipmentManager as the OnEquipmentChanged method in this class.
- output: None
- exception: None

OnEquipmentChanged(EquipmentItem newItem, EquipmentItem oldItem):

- translation: Adds the defence and attack modifiers of the newItem to the armour and damage stat respectively. Removes the defence and attack modifiers of the oldItem to the armour and damage stat respectively.
- output: None
- exception: None

Eat(ConsumableItem consumable):

- translation: Adds the health modifier of the consumable item to the curHealth.
- output: None
- exception: None

Die():

- translation: Calls the base Die() method from CharacterStats and resets the scene.

- output: None
- exception: None

MeleeSound():

- translation: Plays the axeSound or punchSound at the transform of the player depending on whether the equipmentManager says the axe is equipped or not.
- output: None
- exception: None

M17 ZombieStats

Template Module

ZombieStats

Uses

M15

Syntax

Exported Types

~~None~~ Animator, GameObject, Transform

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
Die			
DropItem	Vector3, $\mathbb{N}$		
IsDead		$\mathbb{B}$	

Semantics

State Variables

~~None~~
animator: Animator
zombie: Transform
delay: $\mathbb{R}$
timeAnimStarted: $\mathbb{R}$
isDead: $\mathbb{B}$
isDropped: $\mathbb{B}$
drops: GameObject[ ]

State Invariant

None

Assumptions

This module inherits from CharacterStats.

Access Routine Semantics

Start():

- translation: Initializes the animator, initial HP as well loads the prefabs for each drop and stores them in the drops array.
- output: None
- exception: None

Update():

- translation: Destroys the zombie game object and spawns a random drop item once the isDead boolean becomes true and enough time has passed for the death animation to finish.
- output: None
- exception: None

Die():

- translation: Calls the base Die() method from CharacterStats ~~and destroys the zombie game object this script is attached to~~, sets the isDead boolean to true, and starts the animation/animation timer.
- output: None
- exception: None

DropItem(Vector3 destination, int dropChoice):

- translation: If an item has not yet been dropped, instantiate an item at the specified destination based on the dropChoice parameter
- output: None
- exception: None

IsDead():

- translation: A public method to return the isDead boolean.
- output: isDead
- exception: None

M18 Stat

Template Module

Stat

Uses

None

Syntax

Exported Types

List

Exported Access Programs

Routine name	In	Out	Exceptions
GetValue		$\mathbb{N}$	
AddToStat	$\mathbb{N}$		
RemoveFromStat	$\mathbb{N}$		

Semantics

State Variables

initialValue: $\mathbb{N}$
statChanges: $List < int >$

State Invariant

None

Assumptions

None

Access Routine Semantics

GetValue():

- translation: Initalizes finalValue as the initialValue then adds each statChange in the list to the finalValue to apply the modifiers from equipment and weapons
- output: $out := finalValue$
- exception: None

AddToStat(int stat):

- translation: Adds the stat parameter to the statChanges list
- output: None
- exception: None

RemoveFromStat(int stat):

- translation: Removes the stat parameter from the statChanges list
- output: None
- exception: None

M19 Zombie

Template Module

Zombie

Uses

M14, M15

Syntax

Exported Types

Animator, NavMeshAgent, Vector3, CharacterCombat, CharacterStats

Exported Access Programs

Routine name	In	Out	Exceptions
Start Update MoveToTarget LookAtTarget RandomMovement onDrawGizmos	Vector3		

Semantics

State Variables

detectRadius: $\mathbb{R}$
returnToSpawnRadius: $\mathbb{R}$
distanceToPlayer: $\mathbb{R}$
distanceToSpawn: $\mathbb{R}$
randomAngle: $\mathbb{R}$
animator: $Animator$
player: $Transform$
agent: $NavMeshAgent$
target: $Vector3$
spawnLocation: $Vector3$
enemyCombat: $CharacterCombat$

playerStats: *CharacterStats*
enemyState: *enum EnemyStates*

State Invariant

detectRadius = 5.0
returnToSpawnRadius = 25.0

Assumptions

None

Access Routine Semantics

Start():

- translation: Initializes the spawnLocation as the initial position of the transform and gets components for each state variable from the zombie that this script is attached to.
- output: None
- exception: None

Update():

- translation: In the Passive state, continuously run RandomMovement() and check if the distanceToPlayer comes within the detectRadius, in which case switch to Attack state. In the Attack state, continously call MoveToTarget() to follow the player and check if the distanceToPlayer comes within stopping distance, in which case play the attack animation and attack the player. Also check if the distanceToSpawn has surpassed the returnToSpawnRadius, in which case make the zombie look at and move back to spawnLocation.
- output: None
- exception: None

MoveToTarget():

- translation: Sets the destination of the navmesh agent to the player's position, and calls LookAtTarget() if the enemy is within stopping position.

- output: None
- exception: None

LookAtTarget(Vector3 destination):

- translation: Rotates the zombie object so it is looking in the direction of the destination parameter.
- output: None
- exception: None

RandomMovement():

- translation: Selects a random angle between [0..2π] and gets the zombie to move a certain distance towards this angle.
- output: None
- exception: None

OnDrawGizmos():

- translation: Draws a blue line from the zombie object to its destination it is walking towards and a red sphere around the zombie's detectRadius.
- output: None
- exception: None

M21 Gun

Template Module

Gun

Uses

Syntax

Exported Types

int, int, GameObject, AudioClip, AudioClip, float, float, Tranform, GameObject, Transform, float

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
shoot			
reload			
createBullet			
getAmmoInClip		$\mathbb{N}$	

Semantics

State Variables

ammoPerClip: $\mathbb{N}$
ammoInClip: $\mathbb{N}$
bullet: *GameObject*
gunShotSound: *AudioClip*
reloadSound: *AudioClip*
bulletSpeed: $\mathbb{R}$
fireRate: $\mathbb{R}$
barrelLocation: $Transform$
cam: *GameObject*
startPoint: $Transform$
startTime: $\mathbb{R}$

State Invariant

ammoPerClip > 0
startTime ≥ 0
ammoInClip $\leq$ ammoPerClip

Assumptions

This module is attached to a GameObject which has a RangedWeapon component attached to it.

Access Routine Semantics

Start():

- translation: This method sets *startTime* to the time of starting of the level and sets *cam* to an object of type *GameObject*. This method then called the *reload*() method, and the following actions then take place:
  barrelLocation := Transform of *Gun*
  startPoint := Transform of *cam*
- output: None
- exception: None

Update():

- translation: This method sets the variable $elapsedTime$ to the currentTime. Then, this method checks if the user has pressed the left mouse button or the 'R' button. If the user has pressed the left mouse button and the value of $elapsedTime$ is more than $fireRate$, this method calls the *shoot* method and resets $elapsedTime$ to 0.
- output: None
- exception: None

shoot():

- translation: This method checks if the value of *ammoInClip* is greater than 0. If yes, this method then calls the *createBullet* method, plays a shooting animation and decreases the value of *ammoInClip* by 1.
  If the value of *ammoInClip* is 0, this method calls the *reload* method.

- output: None
- exception: None

reload():

- translation: This method sets the value of *ammoInClip* to that of *ammoPerClip* and plays the reload sound effect.
- output: None
- exception: None

createBullet():

- translation: This method instantiates a *bullet* GameObject and plays a gunshot sound effect.
- output: None
- exception: None

getAmmoInClip():

- translation: This public method returns the private ammoInClip integer.
- output: ammoInClip
- exception: None

M22 BulletDamage

Template Module

BulletDamage

Uses

M17

Syntax

Exported Types

Trigger

Exported Access Programs

Routine name	In	Out	Exceptions
OnTriggerEnter	Collider		

Semantics

State Variables

damage: $\mathbb{R}$
enemyHit: *ZombieStats*

State Invariant

damage > 0

Assumptions

The object collided with has a "collider" component attached to it.

Access Routine Semantics

OnTriggerEnter(Collider other):

- translation: This method detects if the bullet has collided with an object "other". If the object collided with has a tag "Enemy" then the health for that object is reduced by *damage*. This method then destroys the bullet object.

- output: None
- exception: None

M23 DayLightController

Template Module

DayLightController

Uses

None

Syntax

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
Update			

Semantics

State Variables

rate: $\mathbb{R}$

State Invariant

None

Assumptions

This module is attached to the Directional Light object that controls the lighting system.

Access Routine Semantics

Update():

- translation: Rotates the transform of the directional light according a rate proportional to the rate variable.

- output: None
- exception: None

M24 PlayerUI

Template Module

PlayerUI

Uses

M16, M21

Syntax

Exported Types

PlayerStats, Gun, RectTransform, Text

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
SetStats	PlayerStats		
SetHPFill	$\mathbb{R}$		
SetAmmoAmount	String		

Semantics

State Variables

hpBarFill: RectTransform
ammoLeft: Text
gun: Gun
stats: PlayerStats

State Invariant

None

Assumptions

This module is attached to the PlayerUI canvas.

Access Routine Semantics

Start():

- translation: Calls SetStats() to gain a reference to the player statistics.
- output: None
- exception: None

Update():

- translation: Determines if a gun object exists and continiously updates player health and ammo amounts on the UI.
- output: None
- exception: None

SetStats(PlayerStats playerStats):

- translation: Sets the state stat variable to the input parameter playerStats.
- output: None
- exception: None

SetHPFill(float amount):

- translation: Updates the scale of the red component of the HP bar according to the amount of health left, expressed as a float.
- output: None
- exception: None

SetAmmoAmount(string amount):

- translation: Updates the scale of the text of the ammo counter according to the amount of ammo left, expressed as a string.
- output: None
- exception: None