SE 3XA3: Module Interface Specification
Zombie Survival Kit

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

November 9, 2018

Modules

M1: Hardware-Hiding Module

M2: Item (Component)

M3: ConsumableItem (Component)

M4: EquipmentItem (Component)

M5: EquipmentManager (Manager)

M6: InventoryManager (Manager)

M7: Interactable (Object)

M8: Enemy (Object)

M9: ItemStore (Object)

M10: InteractableController (Character)

M11: EquipmentUI

M12: EquipmentSlotUI

M13: InventoryUI

M14: InventorySlotUI

M15: CharacterCombat (Character)

M16: CharacterStats (Manager)

M17: PlayerStats (Manager)

M18: ZombieStats (Manager)

M19: Stat (Component)

M20: Zombie (Object)

M21: FirstPersonController (Character)

M22: Gun (Object)

M23: BulletDamage (Objects)

M2 Item

Template Module

Item

Uses

M6

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

M3 ConsumableItem

Template Module

ConsumableItem

Uses

M2, M6

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

M4 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

M5 EquipmentManager

Template Module

EquipmentManager

Uses

M2, M5, M12

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			

Semantics

State Variables

equippedItems: array of *EquipmentItem*
inventory: *Inventory*

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) and state variable inventory as an instance of the Inventory module.
- output: None
- exception: None

Update():

- translation: Upon keyboard input of "U", perform UnequipAll().
- output: None
- exception: None

Equip(EquipmentItem newEquipment):

- translation: Stores the newEquipment item into equippedItems[newEquipment.equipSlot].
- output: None
- exception: None

Unequip(int SlotIndex):

- translation: Removes the EquipmentItem in equippedItems[SlotIndex].
- output: None
- exception: None

UnequipAll():

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

M6 InventoryManager

Template Module

Inventory

Uses

M2

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

M7 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 M7.
- 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

M9 ItemStore

Template Module

ItemStore

Uses

M2, M7, M6

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 M7) 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

M10 InteractableController

Template Module

InteractableController

Uses

M7

Syntax

Exported Types

Interactable, RaycastHit, GameObject, Vector3, Physics, Input,

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

M11 EquipmentUI

Template Module

EquipmentUI

Uses

M5, M12

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

M12 EquipmentSlotUI

Template Module

EquipmentSlot

Uses

M4, M5

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

M?? InventoryUI

Template Module

EquipmentUI

Uses

M6, M9

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

M14 InventorySlotUI

Template Module

EquipmentSlot

Uses

M6

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