SE 3XA3: Module Interface Specification
BlockBuilder

Team 28, OAC
Owen McNeil, mcneilo
Christopher DiBussolo, dibussoc
Andrew Lucentini, lucenta

December 5, 2018

This document shows the complete specification for the modules used for running BlockBuilder.

Table 1: Revision History

Date	Developer(s)	Change
November 9, 2018	All members	Pushed Rev0 to repo
November 2, 2018	All members	Creation of Rev 0
December 1, 2018	All members	Begin creating changes for rev1
December 5, 2018	All members	Rev1 complete

Function Module

Module

Function

Uses

N/A

Syntax

Exported Constants

SECTOR_SIZE = 16

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
normalize	$\mathbb{R}, \mathbb{R}, \mathbb{R}$	Set of $\mathbb{R}$	~~None~~ invalid_size
sector	Set of $\mathbb{R}$	Set of $\mathbb{R}$	invalid_size

Semantics

State Variables

None

Environment Variables

None

State Invariant

None

Assumptions

- The mathematical operator \ represents integer division. For example $8 \backslash 5 = 1$.

Access Routine Semantics

normalize(~~x, y, z~~ $position$):

- transition: None
- output: ~~$out := \{round(x), round(y), round(z)\}$~~
  $out := \{round(position[0]), round(position[1]), round(position[2])\}$
- exception: ~~None~~
  $exc := (|position| \neq 3) \Rightarrow invalid\_size$

sectorize($position$):

- transition: None
- output: ~~$out := normalize(position[0] \backslash SECTOR\_SIZE, 0, position[2] \backslash SECTOR\_SIZE)$~~
  $out := (normalize(position)[0] \backslash SECTOR\_SIZE, 0, normalize(position)[2] \backslash SECTOR\_SIZE)$
- exception: $exc := (|position| \neq 3) \Rightarrow invalid\_size$

Local Functions

round: $\mathbb{R} \rightarrow \mathbb{Z}$
round(x) $\equiv$ The value of the real x is rounded to the nearest integer.

Block Module

Module

Block

Uses

N/A

Syntax

Exported Constants

~~SECTOR_SIZE = 16~~ None

Exported Types

GRASS = allFacesCoordinates((1, 0), (0, 1), (0, 0))
SAND = allFacesCoordinates((1, 1), (1, 1), (1, 1))
BRICK = allFacesCoordinates((2, 0), (2, 0), (2, 0))
STONE = allFacesCoordinates((2, 1), (2, 1), (2, 1))
DIRT = allFacesCoordinates((0, 1), (0, 1), (0, 1))
inventoryT = {GRASS, SAND, BRICK, STONE}

Exported Access Programs

Routine name	In	Out	Exceptions
cubeVertices	$\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}_{>0}$	Set of $\mathbb{R}$	nNotPos
textureCoordinate	Set of $\mathbb{R}$	Set of $\mathbb{R}$	invalid_size
allFacesCoordinates	Set of $\mathbb{R}$, Set of $\mathbb{R}$, Set of $\mathbb{R}$	Set of $\mathbb{R}$	invalid_size

Semantics

State Variables

None

Environment Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

cubeVertices(x, y, z, n):

- transition: None
- output: $out := \{x - n, y + n, z - n, x - n, y + n, z + n, x + n, y + n, z + n, x + n, y + n, z - n, x - n, y - n, z - n, x + n, y - n, z - n, x + n, y - n, z + n, x - n, y - n, z + n, x - n, y - n, z - n, x - n, y - n, z + n, x - n, y + n, z + n, x - n, y + n, z - n, x + n, y - n, z + n, x + n, y - n, z - n, x + n, y + n, z - n, x + n, y + n, z + n, x - n, y - n, z + n, x + n, y - n, z + n, x + n, y + n, z + n, x - n, y + n, z + n, x + n, y - n, z - n, x - n, y - n, z - n, x - n, y + n, z - n, x + n, y + n, z - n\}$
- exception: $(n < 0) \Rightarrow$ nNotPos

textureCoordinates(p):

- transition: None
- output: $out := \{p[0] * \frac{1.0}{4}, p[1] * \frac{1.0}{4}, p[0] * \frac{1.0}{4} + \frac{1.0}{4}, p[1] * \frac{1.0}{4}, p[0] * \frac{1.0}{4} + \frac{1.0}{4}, p[1] * \frac{1.0}{4} + \frac{1.0}{4}, p[0] * \frac{1.0}{4}, p[1] * \frac{1.0}{4} + \frac{1.0}{4}\}$
- exception: $exc := (|p| \neq 2) \Rightarrow invalid\_size$

allFacesCoordinates($top, bottom, side$):

- transition: None
- output: $out := \{texCoord(top), texCoord(bottom), texCoord(side), texCoord(side), texCoord(side), texCoord(side)\}$
- exception: $exc := (|top| \neq 2 \vee |bottom| \neq 2 \vee |side| \neq 2) \Rightarrow invalid\_size$

Constants Module

Module

Constants

Uses

None

Syntax

Exported Constants

TICKS_PER_SEC = 60
WALKING_SPEED = 5
FLYING_SPEED = 20
GRAVITY = 20.0
MAX_JUMP_HEIGHT = 1.0
JUMP_SPEED = $\sqrt{(2 * \text{GRAVITY} * \text{MAX\_JUMP\_HEIGHT})}$
TERMINAL_VELOCITY = 50
PLAYER_HEIGHT = 2
TEXTURE_PATH = 'texture.png'

Exported Types

None

Exported Access Programs

None

Semantics

State Variables

None

State Invariant

None

World

Module

World

Uses

Block, Constants, Function

Syntax

Exported Types

World = ?

Exported Constants

None

Exported Access Programs

Routine name	In	Out	Exceptions
World		World	
GenerateWorld			
hitTest	Set of $\mathbb{R}$, Set of $\mathbb{R}$, $\mathbb{Z}$	Set of $\mathbb{R}$	invalid_Distance
exposed	Set of $\mathbb{R}$	$\mathbb{B}$	
addBlock	Set of $\mathbb{R}$, Set of $\mathbb{R}$, inventoryT		
removeBlock	Set of $\mathbb{R}$		
showBlock	Set of $\mathbb{R}$		
hideBlock	Set of $\mathbb{R}$		
checkSurrounding	Set of $\mathbb{R}$		
showSector	Set of $\mathbb{R}$		
hideSector	Set of $\mathbb{R}$		
changeSector	Set of $\mathbb{R}$, Set of $\mathbb{R}$		

Semantics

State Variables

blockSet: Set of ((Set of $\mathbb{R}$) × $inventoryT$)
# A set representing all of the blocks in the world at a given position with a given inventoryT.

shownBlocks: Set of ((Set of $\mathbb{R}$) × $inventoryT$)
# A set representing all of the blocks in the world that are visable to the player at a given position with a given inventoryT.

sectors: Set of ((Set of $\mathbb{R}$)× (Set of $\mathbb{R}$))
# Mapping from sector to a list of positions inside that sector.

Environment Variables

None

State Invariant

None

Assumptions

- The constructor Window is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.
- The generateWorld() access routine is called after World() but before any other access routine.
- The showBlock(position) access routine assumes the block at position has already been added to the world with addBlock.
- All Set of $\mathbb{R}$ defined as any inputs or outputs to access routines have a length of 3.
- The operator / represents set difference. I.e. s := s / x means the set s becomes s with the element x removed.

Access Routine Semantics

World():

- transition: $blockSet, shownBlocks := \{\}, \{\}$
- output: $out := self$
- exception: None

generateWorld():

- transition: $blockSet :=$ Set of randomly generated, life-like landforms using elements from inventoryT.
- ouput := None
- exception: None

hitTest(pos, vec, distance):

- transition: None
- output: $out :=$ *A set of* $\mathbb{R}$ *representing the position of a block if it is intersected with the player's line of sight and vector, and is less than distance blocks away.*
- exception: $exc := (distance < 0) \Rightarrow invalid\_Distance$

exposed(position):

- transition: None
- output: $out :=$ *True if one of the faces from the block at position does not exist in the blockSet, otherwise false*
- exception: None

addBlock(position, playerPos, texture):

- transition: $(< (sectorize(position), texture) >\notin blockSet \land position \neq playerPos) \Rightarrow blockSet := blockSet|| < (sectorize(position), texture) >$

- output: None
- exception: None

removeBlock(position):

- transition: $(<(sectorize(position), texture)>\in blockSet)) \Rightarrow blockSet := blockSet/<(sectorize(position), texture)>$
- output: None
- exception: None

showBlock(position):

- transition: (The block at position can be seen by the player) $\Rightarrow (shownBlocks := shownBlocks|| < sectorize(position), getTextureFromSet(sectorize(position)) >$
- output: None
- exception: None

hideBlock(position):

- transition: (The block at position cannot be seen by the player) $\Rightarrow (shownBlocks := shownBlocks/ < sectorize(position), getTextureFromSet(sectorize(position)) >$
- output: None
- exception: None

checkSurrounding(position):

- transition: Check all blocks surrounding ‘position‘ and ensure their visual state is current. This means hiding blocks that are not exposed and ensuring that all exposed blocks are shown. Usually used after a block is added or removed. The routine will call showBlock and hideBlock accordingly.
- output: None
- exception: None

showSector(sector):

- transition: Ensure all blocks in the given sector that should be shown are drawn using addBlock.
- output: None
- exception: None

hideSector(sector):

- transition: Ensure all blocks in the given sector that should be shown are drawn using addBlock.
- output: None
- exception: None

changeSector(before, after):

- transition: Move from sector before to sector after.
- output: None
- exception: None

Local Functions

getTextureFromSet: Set of $\mathbb{R} \rightarrow inventoryT$
getTextureFromSet(p) $\equiv$ The texture in the set blockSet corresponding to the element with position equivalient to p.

Window Module

Module

Window

Uses

World, Block, Constants, Function

Syntax

Exported Types

Window = ?

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
Window		Window	
setExclusiveMouse	$\mathbb{B}$		
getSightVector		Set of $\mathbb{R}$	
getMotionVector		Set of $\mathbb{R}$	
Collision	Set of $\mathbb{R}, \mathbb{Z}$	Set of $\mathbb{R}$	
on_mouse_press	Keyboard Mouse click		
on_mouse_motion	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$		
on_key_press	keyInput		
on_key_release	keyInput		
draw			

Semantics

State Variables

Exclusive: $\mathbb{B}$ *#Determines if the mouse is captured by the window*
Flying: $\mathbb{B}$ *#Determines if flying mode is on/off*
Strafe: Set of $\mathbb{Z}$ *#Determines the direction of movement*
Position: Set of $\mathbb{R}$ *#Defines the player's position in the world*
Rotation: Set of $\mathbb{R}$ *#Defines the relative position of the screen*
Sector: Set of $\mathbb{Z}$ *#An integer list of sectors*
Reticle: Generated Pyglet Graphics

dy: $\mathbb{R}$ *#Defines the relative y velocity of the screen*
Inventory: {GRASS, SAND, BRICK, STONE} *#Set of blocks able to be placed by user*
Block: inventoryT *#The current block being used by the player*
World: World() *#A world object*
Label: Generated Pyglet Label

Environment Variables

keyInput: { "key._W", "key._S", "key._A", "key._D", "key._SPACE", "key._ESCAPE", "key._TAB", "key._1", "key._2", "key._3" } *#The set of keys corresponding to the keys on a keyboard with their respective names.*

leftClick : *#A left click provided by a mouse/track pad*

rightClick: *#A right click provided by a mouse/track pad*

cursorX : $\mathbb{R}$ *#The speed at which the mouse is moving in the x direction (negative for left direction and positive for right direction*

cursorY: $\mathbb{R}$ *#The speed at which the mouse is moving in the y direction (negative for downward direction and positive for upward direction)*

TEXTURE_PATH: *#The path to the image used to load the textures.*

State Invariant

$|Strafe| = 2 \wedge |Position| = 3 \wedge |Rotation| = 2 \wedge |Sector| = 3$

Assumptions

- The constructor Window is called for each object instance before any other access routine is called for that object. The constructor cannot be called on an existing object.
- All access routines except for Window() and setExclusiveMouse(excl) are called by pyglet library TICKS_PER_SEC times a second. The access routines on_mouse_press,

on_mouse_motion, on_key_press, and on_key_release are required for the pyglet library to read user input.

- It is assumed that a 3D envorinment is generated with the pyglet library when Window() is called. The window acts as the player point of view and has a position in the Window given by the set of 3 $\mathbb{R}$ written {x, y, z}.

Access Routine Semantics

Window():

- output: *out* := A window with a default size (defined by the pyglet library) is created on the computer.
- transition: Exclusive, Flying, Strafe, Position, Rotation, Sector, Reticle, dy, Block := False, False, {0 , 0}, {0, 0, 0}, {0, 0}, None, None, 0, Inventory[0]
- exception: None

setExclusiveMouse(*excl*):

- output: None
- transition: $Exclusive := excl$
  i.e The mouser cursor disapears and the pyglet window has exclusive access to the mouse if excl is true
- exception: None

getSightVector():

- output: $out := \{cos(\frac{(Rotation[0]-90)*\pi}{180})*cos(\frac{Rotation[1]*\pi}{180}), sin(\frac{Rotation[1]*\pi}{180}), sin(\frac{(Rotation[0]-90)*\pi}{180})* cos(\frac{Rotation[1]*\pi}{180})\}$ *i.e get the world coordinates of where the camera is looking*
- transition: None
- exception: None

getMotionVector():

- output: *out := The current motion vector of the screen is outputted as a set of three* $\mathbb{R}$ *labelled* $\{x, y, z\}$*, where each element represents the camera velocity in the x, y and z directions respectively.*
- transition: None
- exception: None

Collision(position, height):

- output: None
- transition: *Position := Given the player position (*$\{x, y, z\}$*) and PLAYER_HEIGHT height, a new* $\{x, y, z\}$ *position is calculated after taking into account any collisions with blocks existing in the world. A player cannot move into the square space defined by a block.*
- exception: None

on_mouse_press(*button*):

- output: None
- transition:

	$World :=$
$button = rightClick \wedge World.hitTest(position, getSightVector()) \neq NULL \wedge Exclusive = True$	$World.addBlock(getSightVector(), Block)$
$button = leftClick \wedge World.hitTest(position, getSightVector()) \neq NULL \wedge Exclusive = True$	$World.removeBlock(getSightVector())$
$Exclusive = False \wedge (button = rightClick \vee button = leftClick)$	$setExclusiveMouse(True)$

- exception: None

on_mouse_motion(x, y, dx, dy):

- output: None

- transition: $(Exclusive = True) \Rightarrow$
  $Rotation := (x + cursorX * 0.2, min(max(-90, y + cursorY * 0.2), 90))$
  Note: x and y are the position of the mouse on the screen, if Exclusive = True, these values are the center of the window.

- exception: None

on_key_press($symbol$):

- output: None

- transition:

	$transistion$
$symbol = key.\_W$	$Strafe[0] := Strafe[0] - 1$
$symbol = key.\_S$	$Strafe[0] := Strafe[0] + 1$
$symbol = key.\_A$	$Strafe[0] := Strafe[1] - 1$
$symbol = key.\_D$	$Strafe[0] := Strafe[1] + 1$
$symbol = key.\_SPACE$	$(dy = 0 \wedge Flying = False) \Rightarrow dy = JUMP\_SPEED$
$symbol = key.\_ESCAPE$	$Exclusive := False$
$symbol = key.\_TAB$	$Flying := \neg Flying$
$symbol = key.\_1$	$Block := GRASS$
$symbol = key.\_2$	$Block := SAND$
$symbol = key.\_3$	$Block := BRICK$
$symbol = key.\_4$	$Block := STONE$

- exception: None

on_key_release($symbol$):

- output: None

- transition:

	$strafe[0] :=$
$symbol = key.\_W$	$strafe[0] + 1$
$symbol = key.\_S$	$strafe[0] - 1$
$symbol = key.\_A$	$strafe[1] + 1$
$symbol = key.\_D$	$strafe[1] - 1$

- exception: None

draw():

- output: None
- transition: *Draw the rectile, designated labels, and all block textures provided by the coordinates defined in each element from inventoryT read from TEXTURE_PATH. Only the faces of the squares defined in the shown variable in the World variable are drawn.*
- exception: None