SE 3XA3: Development Plan
Asteroids War Game

Team 12, Team Name: 3XA3 Lab3 Group 12
Tianzheng Mai and mait6
Junhong Chen and chenj297
Eric Thai and thaie1
Linqi Jiang and jiangl21

Table 1: Revision History

Date	**Developer(s)**	**Change**
March 5th,2021	Linqi Jiang	MIS Revision 0
April 11th,2021	Linqi Jiang	Final Revisit MIS
April 12th,2021	Tianzheng Mai	Final Revisit MIS

Matrix Module

Interface Module

Matrix

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Matrix	~~rows~~ R,~~column~~ R	seq of ~~$\mathbb{T}$~~ R	
configure	~~rot~~ R, ~~scale~~ R, ~~transx~~ R, ~~transy~~	set of $\mathbb{R}$	
set	seq of $\mathbb{T}$		
multiply		seq of $\mathbb{R}$	

Semantics

State Variables

$row : \mathbb{R} \sharp therowoftheMatrix$
$column : \mathbb{R} \sharp thecolumnoftheMatrix$

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new Matrix($row, column$):

- output: $out := data[row][column]$
- exception: None

configure($rot, scale, transx, transy$):

- output: $out :=$ $set(cos(rot*\pi)/180*scale, -sin(rot*\pi)/180*scale, transx, sin(rot*\pi)/180*scale, cos(rot*\pi)/180*scale, transy)$
- exception: none

set($row, column$):

- transition: ~~data[row][column] = +i~~ +k — k ∈ R— data[row][column]
- output:None
- exception: None

~~set~~ multiply($row, column$):

- output:$out := (+i|data[i][j] * argument[j] : i = |data|)$
- exception: None

Sprite Module

Interface Module

Sprite

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Sprite	~~name~~ String, ~~points~~ R	seq of $\mathbb{S}$, seq of $\mathbb{R}$	
run	~~delta~~ float		
move	~~seq of~~ float		
updateGrid			
configureTransform			
draw			
findCollisionCanidates		seq of $\mathbb{S}$	
checkCollisionsAgainst	~~canidates~~		
checkCollision	other	seq of $\mathbb{B}$	
pointInPolygon	x,y	seq of $\mathbb{R}$	
collision			
die			
transformedPoints		seq of array	
isClear		seq of $\mathbb{B}$	
wrapPostMove			

Semantics

State Variables

$children : Set \sharp thechildrenoftheSet$
$visible : \mathbb{B} \sharp Whetherthefieldcanbeviewedbyuser$
$reap : \mathbb{B} \sharp reapoftheSprite$
$bridgesH : \mathbb{B} \sharp theconnectionoftheshipheight$
$bridgesV : \mathbb{B} \sharp theconnectionoftheshipWidth$
$collidesWith : Set \sharp howtheshipcollidewithasteriod$
$x : \mathbb{N} \sharp defaultxvalueoftheSprite$
$y : \mathbb{N} \sharp defaultyvalueoftheSprite$
$rot : \mathbb{N} \sharp rotofthegamefield$
$scale : \mathbb{N} \sharp sizeofthegamefield$
$currentNode : \mathbb{T} \sharp representationoftheasteriod$
$nextSprite : \mathbb{T} \sharp thenextSpriteobject$
$preMove : \mathbb{T} \sharp theintendmoveoftheobject$
$postMove : \mathbb{T} \sharp theganurannteemoveoftheobject$

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new Sprite($name, points$):

- output: $out := name, points$
- exception: None

run($delta$):

- transition: $x, y = currentNode.dupe.horizontal, currentNode.dupe.vertical$
- exception: none

move($delta$):

- transition: ~~rot += 360 —— rot -= 360~~ rot > 350 $\implies$ rot - 360 — rot < 360 $\implies$ rot + 360
- exception: none

updateGrid():

- transition: $gridx, gridy = x/GRID\_SIZE, y/GRID\_SIZE$
- exception: None

configureTransform():

- ~~transition: rad = (rot * π)/180~~
- transition : context.rotate, context.scale((rot * π)/180), context.translate
- exception: None

findCollisionCanidates():

- ~~output:$out := canidates$~~
- transition: (nextSprite $\implies$ candiates.push)
  (north.nextSprite $\implies$ candiates.push)
  (south.nextSprite $\implies$ candiates.push)
  (east.nextSprite $\implies$ candiates.push)
  (west.nextSprite $\implies$ candiates.push)
  (north.east.nextSprite $\implies$ candiates.push)
  (north.west.nextSprite $\implies$ candiates.push)
  (south.east.nextSprite $\implies$ candiates.push)
  (south.west.nextSprite $\implies$ candiates.push)
- exception: None

checkCollisionsAgainst(canidates):

- output:$out := canidates$
- exception: None

checkCollision(other):

- transition: $trans, px, py, count = transformedPoints(), trans[i * 2], trans[i * 2 + 1], trans.length/2$

- exception: None

pointInPolygon(x, y):

- output:$out := oddNodes$
- exception: None

die():

- output:$out := oddNodes$
- exception: None

transformedPoints():

- output:$out := trans$
- exception: None

isClear():

- output:out := isEmpty(this.collidesWith) & north.isEmpty(this.collidesWith) & south.isEmpty(this.collidesWith) & east.isEmpty(this.collidesWith) & west.isEmpty(this.collidesWith) & north.east.isEmpty(this.collidesWith) & north.west.isEmpty(this.collidesWith) & south.east.isEmpty(this.collidesWith) & south.west.isEmpty(this.collidesWith)
- exception: None

wrapPostMove():

- transition: $x, y = canvasWidth, canvasHeight$
- exception: None

Ship Module

Interface Module

Ship,SFX,FSM

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Ship	String , sep of R	Ship	
collidesWith		seq of String	
premove	~~delta~~ float		
collision	~~other~~ float		

Semantics

State Variables

None

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new Ship():

- output: $out := (\text{"}ship\text{"}, \underline{[-5,4,0,-12,5,4]}[0,-20,-8,-10,-8,-8,-16,3,-8,0,-8,-8,-8,5,0,7,8$

- exception: None

collidesWith():

- output: $out := ([\text{"}asteroid\text{"}, \text{"}bigalian\text{"}, \text{"}alieanbullet\text{"}])$

- exception: None

premove(delta):

- transition: $(KEYSTATUS.left = True \implies vel.rot = 6 | KEYSATUS.right = True \implies vel.rot = -6 | vel.rot = 0)$
  $(KEYSTATUS.up = True \implies acc.x = 0.5 * cos(rad) \wedge acc.y = 0.5 * sin(rad) \wedge children.exhaust.visible = random() > 0.1 | acc.x = 0 \wedge acc.y = 0 \wedge children.exhuast.visible = False)$
  where rad = (rot - 90) * π / 180
  $(bulletCounetr¿0 \implies buuletCounter = bulletCounter - delta)$
  $(KEYSTATUS.space \implies (bulletCounter <= 0 \implies bulletCounter = 10 \wedge x = x + vectorx * 4 \wedge y = y + vector * 4 \wedge vel.x = 6 * vectorx + vel.x \wedge vel.y = 6 * vertory + vel.y \wedge visible = True))$
  where rad = (rot - 90) * π / 180 , vector x = cos(rad),vector y = sin(rad)
  $\sqrt{vel.x * vel.x + vel.y * vel.y} > 8 \implies$ vel.x = 8 $\wedge$ vel.y =8

- output: None

- exception: None

collison(other):

- transition: $callfunctionSFX.explosion()andGame.explosionAt(other.x, other.y)$
  $Game.FSM.state, visble, currentNode =' playerdied', false, null$
  Finally call Game.live to make sure the game still in progress.

- output: None

- exception: None

BigAlien Module

Interface Module

Ship,Sprite

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new BigAlien	String, sep of R	BigAlien	
top		Sprite	
bottom		Sprite	
setup		newPosition	
preMove	delta		
postmove	y		
Bulletcounter	bullets		
colloides with	String		

Semantics

State Variables

None

State Invariant

~~None~~ children: set $\sharp\, the children of the Big Alien$

Assumptions

The arguments provided to the access programs will be of the correct type.children.top is assigned to be a Sprite type variable and set tobe visible and so is the children.bottom

Access Routine Semantics

new BigAlien():

- output: $out := (''bigalien'', [-20, 0, -12, -4, 12, -4, 20, 0, 12, 4, -12, 4, -20, 0, 20, 0])$
- exception: None

top():

- output: $out := (''bigalien'', [-8, -4, -6, -6, 6, -6, 8, -4])$
- exception: None

bottom():

- output: $out := (''bigalien'', [8, 4, 6, 6, -6, 6, -8, 4])$
- exception: None

collidewith():

- output: [”asteroid”, ”ship”, ”bullet”]
- exception: None

setup():

- output: $newPosition()$
- exception: None

premove():

- transition: $(topCount = 0 \implies topCount + 1)$
  $(bottomCount = 0 \implies bottomCount + 1)$
  $(topCount¿bottomCount \implies vel.y = 1|randomnum < 0.01)$

- output: None

- exception: $cn = 0 \implies None$

bulletCounter():

- output := $bullet.x, bullet.y, bullet.vel.x, bullet.vel.y, visible = x, y, 6 * vectorx, 6 * vetory, true$
  $SFX().laser$
  where rad = 2 * π * random
  where vectorx = cos(rad)
  where vectory = sin(rad)

- exception: None

Local Function

newPosition: int $\implies$ int
random() $<$ 0.5 $\implies$ x = -20 — random() $>$ 0.5 $\implies$ x = Game.canvasWidth + 20
random() is a random math number

Bullet Module

Interface Module

Ship,Sprite

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new Bullet	String, set of R	Bullet	
draw	visible		
premove	~~delta~~ float		
collision	~~other~~ float		
configureTransform			
transformedPoints	float		

Semantics

State Variables

None

State Invariant

~~None~~
time : R $\sharp$ $timeofthebulletexist$
bridgeH: B $\sharp$ $connectionofthebulletheight$
bridgeV: B $\sharp$ $connectionofthebulletwidth$

postMove: wrapPostMove() $\sharp$ *postmove*

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new Bulllet():

- transition:$time, bridgesH, bridgesV, postMove = 0, false, false$
- output: $out :=$ ("*bullet*", $[0, 0]$)
- exception: None

draw():

- transition:$lineWidth, strokeStyle = 15,$ "$FF0000$"
  call save(), beginPath(),moveTo(x-1,y-1),lineTo(x+1,y+1),moveTo(x+1, y-1),lineTo(x-1,y+1),stroke(),restore();
- output : None
- exception : None

preMove(delta):

- transition:$(visible = True \implies time + delta | time > 50 \implies visible = false \wedge time = 0)$
- output: None
- exception: None

collision(other):

- transition:$time, visible = 0, false$
  call currentNode.leave(), currentNode
- output: None
- exception: None

configuretansform():

- transition:None
- output: None
- exception: None

transformedPoints(other):

- transition:None
- output: [this.x,this.y]
- exception: None

AlienBullet Module

Interface Module

AlienBullet

Uses

Bullet

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Exceptions
new AlienBullet	String	seq of $\mathbb{S}$	
draw			

Semantics

State Variables

none

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new AlienBullet():

- output: $out := "alientbullet"$
- exception: None

draw(~~*delta*~~):

- ~~transition: $lineWidth, strokeStyle = 2,'FFA07A'$~~
- transition: visible $\implies$ context.save()
  context.lineWidth = 2
  context.beginPath()
  context.strokeStyle='00FFFF'
  context.moveTo(this.x, this.y)
  context.lineTo(this.x-this.vel.x, this.y-this.vel.y)
  context.stroke()
  context.restore()
- exception: none

Asteroid Module

Interface Module

Asteroid

Uses

Sprite

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	In	Out
new Sprite	"asteroid",[-10, 0, -5, 7, -3, 4, 1, 10, 5, 4, 10, 0, 5, -6, 2, -10, -4, -10, -4, -5]	seq of $\mathbb{S}$,
collision	other	

Semantics

State Variables

$visible : \mathbb{B}$
$scale : 6$
$postMove : \mathbb{T}$
$collidesWith : array$

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

new Asteroid():

- output: $out := "asteroid", [-10, 0, -5, 7, -3, 4, 1, 10, 5, 4, 10, 0, 5, -6, 2, -10, -4, -10, -4, -5]$
- exception: None

collision($other$):

- transition: $scale, vel.x, vel.y, vel.rot = scale/3, random() * 6 - 3, random() * 6 - 3, random() * 2 - 1$
- exception: none

Explosion Module

Interface Module

Ship,Sprite

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
new Explosion	String	Explosion	
lines		seq of R	
draw	B		
preMove	R,R		

Semantics

State Variables

~~None~~ bridgesH:B
bridgesV:B

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

lines(~~other~~):

- transition:$lines.push([x, y, x * 2, y * 2])$
  where x is cos(rad) and y is sin(rad) ard is 2 * π * random
  call currentNode.leave(), currentNode

- output: None

- exception: None

draw(~~scale~~visible):

- ~~transition:$lineWidth = 1.0/scale, strokeStyle =$ "$B22222$" call save(),beginPath(),stroke() and restore()~~

- transition : visible $\implies$ context.save()
  context.lineWidth = 1,0/scale
  context.beginPath()
  context.strokeStyle = "B22222"
  context.stroke()

- exception: None

preMove(delta):

- transition:$(visible = True \implies scale + delta | scale > 9 \implies die())$

- output: None

- exception: None

GridNode

Interface Module

Ship,Sprite

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
enter	sprite	sprite	
leave	sprite		
eachSprite	sprite, other		
isEmpty	int[]	boolean	

Semantics

State Variables

~~None~~ north : null $\sharp$ $northdirectionoftheGrid$
sorth : null $\sharp$ $sorthdirectionoftheGrid$
east : null $\sharp$ $eastdirectionoftheGrid$
west : null $\sharp$ $westdirectionoftheGrid$

State Invariant

None

Assumptions

The arguments provided to the access programs will be of the correct type.

Access Routine Semantics

enter(sprite):

- transition:$nextSprite = sprite.nextSprite$
- output: nextSprite
- exception: None

leave(sprite):

- transition: $ref \wedge (ref.nextSprite! = sprite) \implies ref.nextSprite$ call save(),beginPath(),stroke() and restore()
- output: None
- exception: None

eachSprite(sprite, callback)

- transition:$(ref.nextSprite! = null \implies callback.call(sprite, ref))$
- output: None
- exception: None

isEmpty(collidables)

- transition:$(empty! = ref.visible \vee collidables.indexOf(ref.name) == -1 \implies empty)$
- output: empty
- exception: None