Mobile Games Development 1 Coursework: Code Explanation

BSc Computer Games (Software Development)

Dawid Piotr Kubiak

S1717551

GitHub Repository:

https://github.com/dejwkubikson/Mobile-Games-1-Coursework

I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award.

Dawid Kubiak

Contents

Cookies.js

Functions

saveCookie(name,value)

The functions saves a local cookie on users computer. The name parameter is used as the cookie name and the value parameter is used as the value of the cookie. The cookie's expiry date is set a year after.

readCookie(name)

The functions reads a cookie value for the given name parameter. All cookies are decoded and split into an array. A for loop is used to get the cookie, and as long as the value of the first character is empty, the first character is removed using substring(1) function. Then an if statement is used to check if the cookie name is at 0 index, and if so, it's value is returned by removing the cookie name from the whole cookie string. If no match is found then the function returns a 0.

Drawing.js

Variables

Most of the variables are Image() objects. There are also Array() objects which consist of Image() objects. Furthermore, three more variables are specified for a money shake animation. These are straightforward and will not be listed due to their number.

shakeMoneyTimer

Used in purpose of storing elapsed time the shake animation is running for.

shakeMoney

Holds a Boolean value. Used to determine whether the shake animation should be played.

shakeMoneyPositive

In order to reduce amount of code, this Boolean variable is used to determine if the money is shaken due to positive or negative impact on it. For example, if the player lost money, this would be set to false, however, if the person gained money, this would be set to true.

Functions

drawRemoveConfirmation()

Responsible for drawing a pop-up box for the player to confirm if he is sure that he wants to remove the selected object. First, the function checks if the showRemoveConfirmation value is true, as this would indicate that above needs to be shown. If the object to remove is not null or undefined, the function will move onto the step of drawing the pop-up background image. Then an 'Are you sure…' text is drawn, the object that is to be removed, another text stating the price of removal and finally Cancel and Confirm buttons.

drawObjectRange()

The function is in charge of drawing selected object's range on the map. First, the function checks if an object has been selected by checking if selectedMapObject value is not null or undefined. Since the range can be drawn only for player objects, a statement is used to make sure the object is not a type of an obstacle. Finally, the range will not be displayed in case the user wants to remove the

object. The size is determined by getting object's range, multiplying it by the tiles' size and by two – the two multiplier is used as the range is determined from the middle of the object and needs to cover every direction. The centre X and Y coordinates of the whole range are determined and a range image is drawn.

drawMap()
Draws the map of the game (background).

drawMainMenu()
Draws the main menu image and Start and Instructions buttons.

drawInstructions(pageNo)
Responsible for drawing the instructions screen. The background image is determined by the current page number which is then selected from the instructionImages array. The image is drawn, and depending on current page Next, Next and Previous or Start and Previous buttons are drawn.

drawEndScreen()
The function draws the end screen of the game. All enemies are removed and for aesthetic reasons, the obstacles and player objects are drawn as they were during the gameplay. An overlay, shadow, image and Game Over text are draw on top of that. Player's game statistics are displayed using white (#FFF) 30 pixels Oswald font in the centre of the screen. If players best has not yet been saved, it is now, using the savePlayersBest(time, wave) function located in the player.js file. Then the player's best time and wave is displayed along with a Restart button.

drawSideMenu()
In charge of drawing the side menu. First, the function checks if the menu is expanded. If not, the menu expand button is drawn in the top right corner. If the menu is expanded, the side menu and hide menu button images are drawn. Furthermore, if a menu object is selected, a select image is drawn.

shakeMoneyAnimation(type)
Draws the money shake animation. At the first call – this is checked by comparing if the shakeMoneyTimer is set to 0 and shakeMoney is false - the shakeMoney is set to true and the shakeMoneyPositive is set to the parameter's value. During later calls, the function will increment the shakeMoneyTimer by the frame time, and once the value reaches two seconds (2000 milliseconds), the shakeMoney is set to false and shakeMoneyTimer to 0, making it ready for the next shake animation.

drawInfoText()
Displays the information text for the player. If the information holds any text, the infoTextTimer variable is decreased by the frame time value. The text is displayed at the top of the screen, and once the infoTextTimer is equal or lower than zero, the text is removed.

drawPlayerStats()
Responsible for drawing the time, current wave, lives left and money. The gameTimer is converted to mm:ss format and displayed at the top of the screen. If the shake animation is to be played, the font colour is determined by the fact if this is a 'positive' or 'negative' shake. It would be green for the first one, and red for the latter one. Random X axis offset is generated with the value ranging from 1 to 10, then a random 0 or 1 value determines if the above offset is to be a positive or a negative number. Same is done for generating a random Y axis offset. The text is then drawn in the bottom left corner with the offsets added to its position. This creates a shake effect. Otherwise, the money is displayed

in a fixed position. Finally, lives are displayed in the bottom centre of the screen by getting the full width of all lives and drawing them one by one using the life image.

drawReloading(obj)

Draws the reload animation. The current context is saved and then translated to the object's, specified in the parameter, position. The context is rotated by a radian calculated by dividing current reloadTimer by PI value and then by one hundred to make the animation slower. The reload image is drawn and context is restored.

drawObstacles()

Draws the obstacles on the map by iterating through the obstacles array. The images are drawn depending on the obstacle object's position and its image.

drawPlayerObjects()

In charge of drawing player objects. Iterates through the player objects array and, depending if the object run out of ammo, uses its 'empty' or with ammunition image. The base of the object is drawn (this allows to rotate the object without rotating the base), and enemy array is iterated to get enemy positions. If the enemy is withing the range, the object rotates towards and shoots at the enemy. The angle is calculated using getAngleTowardsObject(fromX, fromY, toX, toY) which is found in the game.js file. Shooting is done through shootAtEnemy(obj, enemy) function located in the same file. If there are no enemies within the range, or no enemies at all, the object is drawn with the last saved rotation.

drawBullets()

Draws the bullets that are shot by player's objects. First, the function calls moveBullets() – game.js – and iterates through the bullet array. The context is saved, the bullet rotated towards the enemy, and if the bullet is a type of a rocket, a flame image is also drawn at the bottom of it. The context is then restored.

drawSpecialEffects()

Used to draw special effects such as rocket or tank explosions. A for loop iterates through the rocket explosion array (which holds animation objects), and draws an appropriate image depending on the current frame animation. This image is taken from the rocketExplosionImgArray. Any finished animations are filtered out from the array. Same is done for tank explosions.

drawEnemies()

First, if the game is not paused, moveEnemies() is called, which is stored in the game.js file. Then the function iterates through the enemy objects array, saves the context, moves it to the enemy's position, and depending on current enemy direction, its image is rotated. Finally, the context is restored.

drawProhibited()

The function draws prohibited images by iterating through the prohibitedObjArray. A pulsing effect has been created by modifying the context's globalAlpha value. The value is determined by a sinus of prohibited object's elapsed timer divided by 100 used in order to slower the pulsing effect. The prohibited object and its image is draw for one second, so after the time has elapsed, its filtered out from the above array.

drawGame()

Responsible for executing all drawing functions. Clears the context, calls all drawing functions and fills the context.

drawPause()

Responsible for drawing the pause screen. First, a 'shadow' image, then pause text image and finally the resume button.

Game.js

Variables

Game variables

These variables hold the canvas, its context, current game scene, if the game is paused or not (pasuedGame), single tile size, the time of last executed frame, last frame time and information text.

Gameplay variables

These variables store the current wave, how much money and lives the player has, game time, obstacle remove cost and the amount of money the player gets for killing an enemy.

Wave variables

These variables store how many enemies are to be created, the wave timer, how often the enemies are to be created and if the wave ended.

State variables

These store the current instructions page, selected row in the menu, if the menu is expanded, if a player selected a menu object, if the player selected the obstacle remover option or if the remove confirmation screen should be shown.

Functions

msToMMSS(milliseconds)

The function converts milliseconds to mm:ss time format. First, the milliseconds are converted into seconds. Then minutes are extracted by dividing the seconds by 60 and converting the result into an integer which results in a whole number. The rest of the seconds (not extracted in the above conversion) is gathered using the remainder of seconds and 60. The function also pads zeros on left to minutes and seconds if they are lower than 10. Finally, a string is returned in a mm:ss time format.

getTileDistance(fromX, fromY, toX, toY)

The function returns the distance from one point to another in tiles. Used to determine the tile distance from player object to the enemy as player object's range is determined in tile distance. A Pythagoras theorem is used to get the distance between the objects and the result is divided by the tile size.

getAngleTowardsObject(fromX, fromY, toX, toY)

Returns the angle from one point to another. Used to determine the rotation of player's objects towards the enemy. Math.atan2() is used to get the angle in the plane and then the result is converted from radians to degrees. If from point is on the right side of the to point, then the angle is negative. Finally, the result is multiplied by minus one due to how the image are rotated at the start.

objectBehaviour(obj)

Performs the normal object behaviour. Used only to make the object check if it should reload. If its ammunition is equal to zero, a reload(obj) function is called.

createBullet(obj, enemy)

The above function creates a bullet object. A bullet is created for each barrel an object has. Current rotation of the player object and its position is assigned. Then, an offset is created so that if the

object has two barrels they will not be fired from the same coordinate. If the object shoots rockets, the typeOfAmmo is set to rockets, otherwise to bullets. This is used to determine if special effects, such as rocket explosion, should be created. All appropriate values are assigned to the bullet object and it is added to the bulletArray array.

moveBullets()

The function iterates through the array holding all the bullet objects. The position difference between the bullet and the enemy is calculated, then the angle the bullet should be facing and then the velocities needed to reach the bullet. If the bullet is within a certain distance to the enemy, the bullet's reachedEnemy is set to true. A dealDamage(bullet) function is called and if the bullet is a rocket, a createRocketExplosion(x, y) function is called. The velocities are then added to appropriate X and Y positions of the bullet.

dealDamage(bullet)

Deals damage to the enemy. First, enemy's armour is reduced. Once that reaches zero, the bullet will affect the enemy's health. If the enemy's health reaches zero, its isDead is set to true. Moreover, a 'positive' money shake animation is executed using the shakeMoneyAnimation(true) (drawing.js) method. Finally, the bullet array is then filtered by excluding bullets that reached an enemy and a function removeDeadEnemies() is called.

removeDeadEnemies()

Removes the dead enemies and adds player the money. All dead enemies are filtered from the enemyObjArray and stored in an array named deadArray. Then, the player's money is increased by the above arrays length multiplied by the amount of money the player gets per killed enemy. If the mentioned enemy is a tank, an explosion animation is created and an explosion sound is played. All dead enemies are filtered out from the array.

createTankExplosion(posX, posY)

Creates a tank explosion animation object that holds the X and Y coordinates and frames passed for this animation. This is then added to the tankExplosionsArray.

createRocketExplosion(posX, posY)

Creates a rocket explosion animation object that holds the X and Y coordinates and frames passed for this animation. This is then added to the rocketExplosionArray.

reload(obj)

Performs a reload action on the given player object. The object's reloading value is set to true. This prevents it from shooting while reloading. The reload timer is incremented by the frame time. If the reload time has been reached the object successfully reloaded the ammunition. Otherwise, drawing the reloading image by calling the drawReload(obj) method in drawing.js.

shootAtEnemy(obj, enemy)

Shoots bullets at enemies. First, checking if the object is not reloading, then if the object has enough ammunition. Finally, checking if enough time has passed since last shot. If all above is true, the createBullet(obj, enemy) method is called, ammunition reduced and the object's shot sound is played.

moveEnemies()

Moves the enemies. First, the enemyObjArray is sorted, keeping the enemies closest to the end point first. Then for loop that iterates through the above array, checks if the enemy has reached the next

point. After that, enemy's direction is determined, and appropriate axis manipulated by enemy's speed. Finally, if the enemy has reached the end point, endPointReached() function is called.

endPointReached()
Removes all enemies that reached the end point by filtering the array, reduces player's money and lives, and in case the player lost all the lives, changes the current scene to the end scene. Furthermore, shake money animation function is called.

restartGame()
Restarts the game. Defaults all variables and removes any objects that are not present at the start of the game.

startGame()
Starts the game by creating the obstacles and setting the current scene to "game".

showInstructions()
Changes current screen to instructions. Sets the instructions page to page number one.

endGame()
Changes current scene to "end". This takes the player to the end screen.

update()
Main function, responsible for running the whole game. Depending on current scene, appropriate functions are called. During the game scene, frame times are recorded, drawGame() function (drawing.js) is called and appropriate information is displayed depending on user selection. If the game is paused, the drawPause() method is called and the background music is paused. At the end, the update is called again through requestAnimationFrame.

getEnemyType()
Returns randomly generated enemy type to be created for current wave. This makes each game unique. "Tank-2" and "Soldier-2" enemies are created after wave 4. Random numbers are generated and depending on them, appropriate strings of enemy types are returned.

waitForNextWave(timer)
Waits for all enemies to get killed or reach the end point. Then adds the player money and performs a money shake animation. All player objects are forced to reload to get ready for next wave. Furthermore, the function displays the information text showing the player how much time is left for the next wave. If enough time has elapsed since the end of the wave, variables such as wave and waveEnemyAmount are modified accordingly to the new wave value.

createWave(timer)
Creates enemies in random time differences. If all enemies for that wave have been created, the waveEnd value is set to true.

objects.js

Variables

Object arrays
Various arrays used to store objects.

objToRemove
Used for assigning an object that the player selects to be removed.

selectedMapObject
Used for assigning an object that the player selects on the map. Used to display range.

Functions

checkIfPossibleToBuild(posX, posY)
Checks if it is possible to build on given tile. First, checks if the tile belongs to the path the enemy moves by. Then, checks if the tile is free from any obstacles and player objects. If at any point it is prohibited to build on that tile, an information text is displayed for the player, and a prohibitedObj is added to the prohibitedObjArray that then is used to show prohibited image for the player.

createObjectOnMap(posX, posY, objNumber)
Creates player object on the map. First, checks if it is possible to build on the selected tile by calling the checkIfPossibleToBuild(posX, posY) method. A new Audio() object is created to assign object's shooting sound rather than use the same audio file across all objects. This would result in playing the sound only for one object, as the sound would already be in the play() state. A default player object is created and, depending on the requested objNumber, the values for above object are populated accordingly. After that, the function checks if the player has enough money to build the object. If not, shake money animation is played and information text shown. Otherwise, the money is reduced by the price of the object and the object is added to the playerObjArray. Finally, selectedMenuObject is set to false in order to reset the selected menu object so that the player does not accidentally create it again.

removeObject(objToRemove)
Removes the given object from the map. Checking what type of object it is and filtering it out from appropriate array. Subtracting from player's money the cost of removing the object.

createEnemy(typeToUse)
Creates an enemy object. First, a default object is created, and then, appropriate values assigned depending on which object is requested. The object is then added to the enemyObjArray.

player.js

Variables

playerBestTime
Integer value. Holds the player's longest survival time.

playerBestWave
Integer value. Holds player's longest survival wave.

playerBestSaved
Boolean value. Used to check if player's best has been saved. Prevents from saving each frame.

Functions

savePlayersBest(time, wave)
Saves players best survival time and wave at the end of the game. Calls saveCookie(name, value) function located in the cookies.js file.

assignPlayersBest()

Gets and assigns best survival time and best wave. Used at the beginning of the game. Calls readCookie(name) – cookie.js - and parses the returned value to integer. Finally, if the parse resulted in NaN, zero values are assigned to the playerBestTime and playerBestWave variables.

raycast.js

Functions

getCoords(elID)

Returns the coordinates of given element from the top and left of the document.

getTileCoords(posX, posY)

Returns the zero coordinates of a tile for given position (position where the tile starts). Return values are calculated by flooring the passed parameters after dividing by tile size and then multiplying by the tile size.

touchRaycast(e)

Used to execute events depending on player's touch or click events and current state of the game (e.g. expanded menu). First, X and Y coordinates are gathered by getting the position of touch or click on the canvas. Then, depending on current scene, appropriate buttons or fields are checked. For example, if in game scene, the function will check if the game is paused, if so, the position of touch or click will be compared and checked if the Resume button has been pressed. If the menu is expanded, the function will check if one of the menu objects has been pressed. The functionality extends to checking if an obstacle has been pressed, player object and so on.

setup.js

Variables

currentResource

Integer value, holds the current resource being loaded.

resourcesToLoad

Integer value, holds the amount of resources to load.

spawnPoint

Holds the X and Y coordinates of the starting point and the direction the enemy should move.

endPoint

Holds the X and Y coordinates of the end point and the direction the enemy should move.

movePoints

Holds the X and Y coordinates of all points the enemies need to follow in order to reach the end point.

movementTiles

Holds the X and Y coordinates of all movement tiles. This is used to prevent the player from placing any objects on them.

Functions

createObstacles()

Creates all obstacles on the map by setting up obstaclesArray with obstacle objects.

loadResources()
Used to load resources such as images and sounds

resourceLoad()
Called with every image or sound load. Increments the currentResource variable, and once the resourcesToLoad value is reached update() function is called. This prevents any errors happening due to not loaded images or sounds.

sounds.js

Variables
All variables are used to store Audio() objects.

Functions

playBackgroundMusic()
Plays background music. Toggled on game start or when resuming after pause.

pauseBackgroundMusic()
Pauses background music. Toggled when in pause screen.

Resources

Images
https://www.kenney.nl/assets/tower-defense-top-down

https://opengameart.org/content/top-down-tanks-redux

https://opengameart.org/content/ui-pack

Music
https://www.bensound.com/royalty-free-music/track/funny-song

Sound Effects
https://freesound.org/people/florianreichelt/sounds/459973/

https://freesound.org/people/EFlexMusic/sounds/388528/

https://freesound.org/people/SoundFX.studio/sounds/456272/

https://freesound.org/people/InspectorJ/sounds/328864/

https://freesound.org/people/qubodup/sounds/182429/