Table of Contents

**Main Screen2**

**Help Screen3**

**Game.Java4-6**

Sprite Objects5

Graphics Object5

Draw Clouds Method5

Draw Rain Method……………………………………………………………………………………………………………………………….5

Pause Menu6

Projectile System7

ShotCollision Method7

GroundCollision Method7

TankFire(2,3,4,5) Method7

HitTest(2) Variables7

Weapon States7

FireShot Method7

CreateHole Method7

Topy[] Array7

TankShoot(1,2) Variable7

ResetShot Method7

Wind/Gravity7

KeyPressed Method7

**Main Screen**

Method Name: mainScreen

The main screen is the first screen that the user will see as they load up Tank Wars. It has music that plays and four options to access: Space Bar To Start, (H) Help Screen, (T) Tutorial, and Press Esc To Exit. When changing to a different screen the screen flashes because the game is testing the possible display modes until it finds the right one (If you store this display mode as a variable and call it every other time it will stop the game from flashing when switching again after the first time). By pressing Esc, the key adapter simply calls System.exit(-1); terminating the program. **Help Screen**

**Method Name:** helpScreen

The help screen is the place where we go to for keyboard commands and settings. The screen presents the following static options:

* Turret Movement
* Tank Movement
* Power Adjustment
* Weapon Cycle
* Fire

And the following dynamic options:

* Cycle wind
  + 0 – No Wind
  + 1 – Randomized After Every Shot
  + 2 – Constant Continuous Wind
  + 3 – Completely Randomized
* P1 Tank Color
  + 1 – Green
  + 2 – Red
  + 3 – Blue
  + 4 – Yellow
  + 5 – White
* P2 Tank color
  + 1 – Green
  + 2 – Red
  + 3 – Blue
  + 4 – Yellow
  + 5 - White
* Music Selection
  + Song Titles

And, obviously, Esc returns the user to the main screen.

**Game.Java**

**Sprite Objects**

Contain Animation Objects

Animation Objects contain frames from images (can contain just one frame)

Process:

Initiate the image:

Image cloud3 = loadImage("images/Cloud3.png");

Initiate the animation using the image(s):

Animation cloudSun3 = new Animation();

cloudSun3.addFrame(cloud3, 200);

Animation rainfallAnimation = new Animation();

rainfallAnimation.addFrame(rainImage, 300);

Initiate the Sprite with Animation:

cloudSunny3 = new Sprite(cloudSun3);

rainSprite = new Sprite(rainfallAnimation); //Initializes rainSprite

rainSprite.setY(40); //sets Y-axis rainSprite starting position

rainSprite.setX(1); //sets X-axis rainSprite starting position

**Graphics Object g: The Main Graphics Item.**

G is the generic graphic object used to set the color and draw images. Whenever g is altered in a sequence, the last alteration is set. Strings drawn with g display the color most recently set.

Example - Game Over Menu

Initiate color item:

Color c = new Color(1.0f, 1.0f, 1.0f, 0.6f);

Set color:

g.setColor(c);

Draw image:

g.fillRoundRect(260, 205, 250, 60, 15, 15);

Set new color:

g.setColor(Color.black);

Draw Image:

g.drawImage(BOOM.getImage(), Math.round(player.getX() - 14), Math.round(player.getY() - 75), null);

Draw String:

g.drawString("GAME OVER: PLAYER " + playerNumber + " WINS",300,230);

**Draw Clouds Method**

DrawClouds(Sprite cloudName, Integer windMultiplier)

Draws the clouds in initial positions and updates their position with update command.

Takes a Sprite Image (cloud) and a wind speed multiplier (integer). The windMultiplier simulates distance to the horizon.

**Draw Rain Method**

public void DrawRain(Sprite rainImg, Integer windMultiplier) // windMultiplier is static 70 (determines the sprites setVelocity or rainfall speed)

{

rainImg.setState(1);

//Checks to see if rain image has cycled off screen

if (rainImg.getY()>600 || rainImg.getY()<0)

{

//Resets rain image to 0 on the y axis

rainImg.setY(0);

}

//Conditional: (If windf > 30mph) ~~(Rain mph = Windf\*10000)~~

if (Windf >= .003f)

{

//Set velocity, speed multiplied by a negative to account for proper direction

rainImg.setVelocityY(Math.abs(Windf)\*Math.abs(windMultiplier));

//If wind is < 30mph set rain velocity at static 29mph (.0029f)

}

else

{

//Set velocity, speed multiplied by a negative to account for proper direction

rainImg.setVelocityY(Math.abs(.0029f)\*Math.abs(windMultiplier));

}

}

**Pause Menu (in Game.java)**

The Pause Menu can be toggled during the game by pressing 'O'.

Here you can see P1 and P2 controls

you can change wind settings (WindVar) -- press 'W'

0 - no wind

1 - randomized after every shot

2 - on and constant

3 - completely randomized

you can change Music

- Press M

Also at the bottom you see you key commands such as:

Press Z to change terrain

Press R to restart the game

Press Esc to end the game

Note that normal controls are disabled in Pause Menu.

The Pause Menu is open if PauseMenuOpen = true.

Relevant Code

---------------

Menu is drawn in method DrawMessages()

Pause key controls are found in keyPressed()

\*\* also - creating GameSTATEchanged was necessary in order to be able to

change gameplay mode during the game (see Game.java line 474)

otherwise, it just reverts GameSTATE to whatever it was set to originally.

**Projectile System:**

* ShotCollision-Sees if shot hit tank.
  + Receives: tank, shot, player
  + Creates variables for the corners of the tank and shot. Then determines if the shot image is inside the tank. If it is it returns true else it returns false.
  + Also this method is what makes shot 2 be homing by setting its horizontal velocity to 0
  + Returns: Boolean
* Ground Collision-moves tank to top of ground and sets slant
  + Receives: tank, player
  + Moves the tank so that it is located on the slant below it or above it. This is used for falling and for moving. Moving moves the tank left and right this moves the tank up and down with the land.
  + Returns: void
* Tankfire/Tankfire2/Tankfire3/Tankfire4/Tankfire5-places shot, sets state, and velocity
  + Receives: graphics, player
  + Creates the shot image, places it at the appropriate location, sets the state of the shot to 1 and sets velocity
  + Returns: void
* HitTest/hitTest2
  + If a tank has hit an object this will become 2
* Weapon States(what each state means)
  + State 1 means that the object is in the air/justfired/justcollided
  + State 0 means the shot is not supposed to be displayed
* Fireshot-sees if shot hit ground
  + Receives: Shot, graphic, player, weapon
  + Determines if the shot has hit the ground and if it has calls CreateHole
  + Returns: void
* CreateHole-resets everything and creates an indent
  + Receives: shot weapon player
  + Sets hittest(2)=2. Creates a semi circle whole based on which shot is used.
  + Returns: void
* topy[]
  + Array which holds the top of the Land
* Tankshoot1/tankshoot2 variable
  + >1 shot has not just been shot
  + =1 shot was just shot
* reset shot
  + Receives: Shot, player
  + Resets the state of the shot, hittest, velocity and sets location off screen
  + Returns: shot
* Wind\Gravity
  + Every interaction of Update causes wind and gravity to reduce/increase the velocity of the projectile in the appropriate direction. In this game Acceleration of gravity=.02 It is important to remember that this amount is subtracted from acceleration every second. It is important to remember the equation V=Vo + AT therefore if every second we add A to Vo we will get the velocity for this second.
* Key Pressed-does functions whenever a key is pressed
  + This method just seems to always work. I am not sure how it works but whenever a key is pressed it seems to be sent to this. Most keys work there are some exceptions so testing before using is important to determine if that key works. Overall this method simply takes in a key that is pressed and does a function. To create a turn based game and other pauses in the game instead of removing all functionality from a player a player is simply unable to press a key.