GDD 1200 Project Increment 5

Step 1: Add health display

After: Chapter 14

For this step, you're adding a text display for the burger health.

1. Add code to the `Game1` `LoadContent` method to load the Arial20 font into the `font` variable
2. Add code to the `Game1` `LoadContent` method to set the `healthString` variable to the `GameConstants` `HealthPrefix` constant concatenated with the current burger health (using the burger's `Health` property). Make sure you do this AFTER creating the burger object!
3. Add code to the `Game1` `Update` method to set the `healthString` variable to the `GameConstants` `HealthPrefix` constant concatenated with the current burger health (using the burger's `Health` property). For efficiency, you should ONLY do this whenever the burger takes damage.
4. Add code to the `Game1` `Draw` method to draw the `healthString` at the location given by the `GameConstants` `HealthLocation` constant

When you run your game, you should see the burger's health displayed and modified as the burger takes damage.

Step 2: Add scoring

After: Chapter 14

For this step, you're adding scoring to the game.

1. Add code to the `Game1` `LoadContent` method to set the `scoreString` variable to the `GameConstants` `ScorePrefix` constant concatenated with the current score
2. Add code to the `Game1` `Update` method that adds the value of the `GameConstants` `BearPoints` constant to the score variable when a collision between an active french fries projectile and a teddy bear is detected (you already have a block of code that detects this collision from your previous work). Also, set the `scoreString` variable to the `GameConstants` `ScorePrefix` constant concatenated with the current score
3. Add code to the `Game1` `Draw` method to draw the `scoreString` at the location given by the `GameConstants` `ScoreLocation` constant

When you run your game, you should see the score displayed and modified as you hit teddy bears with french fries.

Step 3: Change burger control scheme

After: Chapter 14

Change the game so the player no longer uses the mouse to control the burger. Make it so WASD are used to move the burger and space is used to make the burger shoot. You'll have to change the second parameter of the `Burger` `Update` method to a `KeyboardState` parameter to make this work properly. The `GameConstants` class contains a `BurgerMovementAmount` constant you should use for the movement part. You do NOT have to avoid the Doom Strafe 40 bug in your code.

Step 4: Add burger and teddy bear shooting sound effects

After: Chapter 15

It's time to start adding sound effects to our game. For this step, you're adding sound effects when the burger or the teddy bears shoot.

1. Add code to the `Game1` `LoadContent` method to load all the sound effects for the game.
2. In the `Game1` `LoadContent` method, change the last argument in your call to the Burger constructor from null to the appropriate sound effect field for the burger shooting sound
3. Add code to the `Burger` `Update` method to play `shootSound` when the burger shoots a projectile
4. In the `Game1` `SpawnBear` method, change the last argument in your call to the TeddyBear constructor from null to the appropriate sound effect field for the teddy bear shooting sound
5. Add code to the `TeddyBear` `Update` method to play `shootSound` when the teddy bear shoots a projectile

When you run your game, you should hear sound effects when the burger and the teddy bears shoot projectiles.

Step 5: Add teddy bear bounce sound effects

After: Chapter 15

For this step, you're adding sound effects when the teddy bears bounce off the sides of the game window or off each other.

1. In the `Game1` `SpawnBear` method, change the second to last argument in your call to the TeddyBear constructor from null to the appropriate sound effect field for the teddy bear bouncing sound
2. Add code to the `TeddyBear` `BounceTopBottom` method to play `bounceSound` when the teddy bear bounces off the top or bottom of the game window
3. Add code to the `TeddyBear` `BounceLeftRight` method to play `bounceSound` when the teddy bear bounces off the left or right of the game window
4. Add code to the `Game1` `Update` method to play the appropriate sound effect field for the teddy bear bouncing sound when two teddy bears collide with each other

When you run your game, you should hear sound effects when the teddy bears bounce off the sides of the game window or off each other.

Step 6: Add burger damage sound effects

After: Chapter 15

For this step, you're adding sound effects whenever the burger take damage.

1. Add code to the `Game1` `Update` method to play the appropriate sound effect field when the burger takes damage from colliding with a teddy bear or a projectile

When you run your game, you should hear sound effects whenever the burger takes damage.

Step 7: Add explosion sound effects

After: Chapter 15

Explosions should also have sound effects. For this step, you're adding sound effects when an explosion is created.

1. Add a parameter to the `Explosion` constructor for a `SoundEffect` that will be played at the end of the constructor. You'll need to add a using statement for the appropriate namespace to get this to compile
2. Play the sound effect parameter at the end of the `Explosion` constructor
3. Add code to the `Game1` `Update` method to pass the appropriate sound effect field as the final argument whenever a new `Explosion` object is created

When you run your game, you should hear sound effects when the explosions start.

Step 8: Add losing sound effects

After: Chapter 15

For this step, you're adding a sound effect when the burger's health reaches 0.

1. Add code to the `Game1` `Update` method to call the `CheckBurgerKill` method whenever the burger takes damage
2. Add code to the `Game1` `CheckBurgerKill` method that checks if the burger's health is 0 (using the burger's `Health` property) and the burger isn't dead yet (using the `Game1` `burgerDead` variable). If both those conditions are true, set the `burgerDead` variable to `true` and play the appropriate sound effect field for a burger death

When you run your game, you should hear a sound effect when the burger's health reaches 0.

Step 9: Celebrate!
That's it − you finished the optional project. Shoot the teddy bears to your heart's content, try to best your high score, and celebrate however you see fit!

TURNING IN YOUR ASSIGNMENT

This portion of the programming assignment is worth 3% of your overall course grade. **Only provide one submission per guild**.

You're required to turn in ALL of the following by the beginning of the scheduled class time on the due date:

Electronic Copy

1. Zip up your entire assignment folder into a file named <yourguildname>.zip. Log into Blackboard and submit the file into the appropriate assignment.

IMPORTANT NOTE: If your zip file doesn't contain all the required files or is zipped up using WinZip, 7Zip, or any other program different from the default Windows compression utility, you'll receive an **AUTOMATIC 0** on this assignment. Since this assignment is worth a good chunk of your overall course grade, I strongly suggest you use the Windows compression utility and check your zip file to make sure it's complete before submitting it.

LATE TURN-INS

- Turn-ins are due at the beginning of the scheduled class time on the specified due date
- No late turn-ins will be accepted