

## Project Increment 5 (without XACT)

These steps assume you're not using XACT to implement the sound effects in the project. Although our company used XACT for the sound effects in all our commercial projects, implementing the game audio using the `SoundEffect` and `SoundEffectInstance` classes (in other words, not using XACT) is more similar to the way things are done in Unity (for example). In addition, I've had students run into more XACT problems in the recent past than I like. I recommend you implement the sound effects in the project without XACT (using these instructions), but if you decide you want to use XACT, use the instructions in the other pdf for the implementation steps.

### **Step 1: Add burger and teddy bear shooting sound effects**

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. Notice that the sound content is contained in a sounds subfolder in the `GameProjectContent` project; you'll need to include that subfolder in your path to the audio content when you load it
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
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 2: Add teddy bear bounce sound effects**

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 3: Add burger damage sound effects**

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 4: Add explosion sound effects**

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 5: Add losing sound effects**

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 6: Add health display**

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 HEALTH_PREFIX` 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 HEALTH_PREFIX` 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 HEALTH_LOCATION` constant

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

#### **Step 7: Add scoring**

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 SCORE_PREFIX` constant concatenated with the current score
2. Add code to the `Game1 Update` method that adds the value of the `GameConstants BEAR_POINTS` 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` `SCORE_PREFIX` 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` `SCORE_LOCATION` constant

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

#### **Step 8: Change burger control scheme**

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 `BURGER_MOVEMENT_AMOUNT` constant you should use for the movement part. You do NOT have to avoid the Doom Strafe 40 bug in your code.

#### **Step 9: Celebrate!**

That's it – you finished the required project. Shoot the teddy bears to your heart's content, try to best your high score, and celebrate however you see fit!