Programming Project #5
CpSc 4160/6160: Data-Driven 2D Game Development
Computer Science Department
Clemson University
(1) Object Pooling, (2) Collisions,
(3) Explosions, and (4) Projectiles
Brian Malloy, PhD
April 5, 2017

Due Date:

To receive credit for this assignment your solution must be submitted, using `handin`, by 8 AM, Friday, April 21st, 2017. You may receive 90% of the grade if you submit within three days of the deadline.

Project Specifications:

The goal of this project is to incorporate more action and interaction into your animation so that it includes projectiles, collisions, and explosions. Your goal should be to develop a consistent theme so that you game begins to tell a story that reaches a conclusion, that is, the player and your game must reach a conclusion. However, keep the scope reasonable, so that you can produce a robust game.

Sprites: For this project, you are not required to draw your own sprites, but you **may not** use any sprites that I have provided. If you use sprites that you did not build yourself, they must be available under the open source license and you must specify, in your **ascii** README, the site where you got the sprites.

Data-Driven: your game must be data driven so that you read game constants from an XML file.

Story Driven: The requirements listed below should guide this project; however, if one or more of the requirements is inconsistent with the theme or story that you are trying to tell, stop by during office hours or send an email with times that we might meet to discuss your game idea and to negotiate a trade for the inconsistent requirement.

Object Pool: However, one requirement that is not negotiable is that this project must include object pooling and your information HUD should illustrate the state of the pool. For example, Figure 1 illustrates, in the upper right corner, the instructor's use of an object pool for bullets, with **3** bullets active and **2** bullets in the pool. This figure is intended only for illustration and should not be interpreted to mean that you must have bullets and they must be pooled in your game. The point is that you must implement an object pool and you must demonstrate your use of the pool in your information HUD. You can pool bullets, explosions, or some other dynamically created object. We will discuss object pooling in lecture and there is a lot of internet information about object pooling; you should consult this information if you need additional help in understanding object pooling.

This project should include requirements from the previous project:

- Either submit an mp4 movie that you generated, or make sure the F4 key works so that the TA can generate a movie for you.
- Include a well-controlled player object, and an animation that creates the illusion of depth.
- Your information HUD should appear at the start of your game and remain long enough to enable the user to see how to use your game. This HUD should be toggled with F1, and provide information

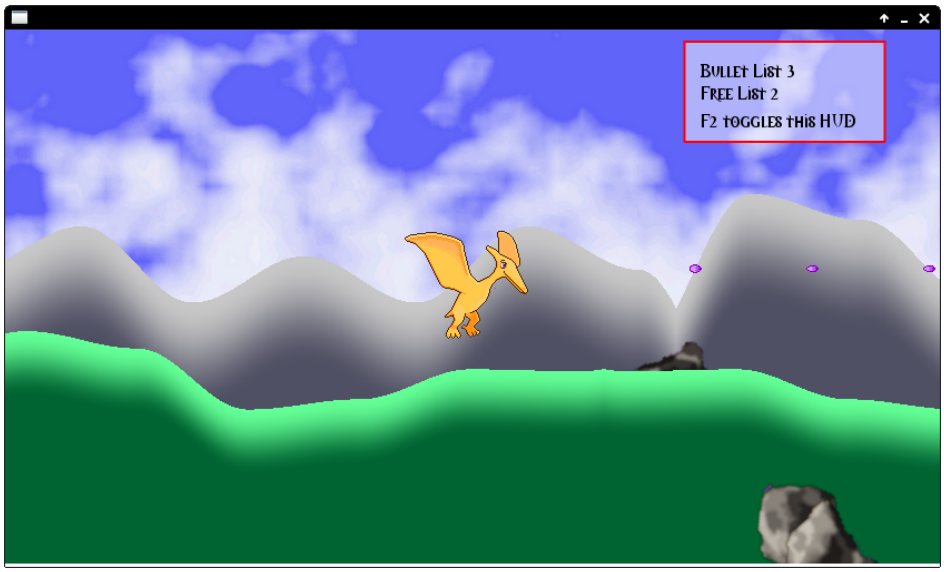

Figure 1: An illustration of object pooling, with three active bullets and one bullet in the "pool"

about how to move the player, and any other game information you wish to display.

The new requirements include:

1. Projectiles: you don't have to shoot bullets, but you need projectiles.
2. Collision detection that triggers explosions; choose a strategy or use all 3.
3. Your player should explode and, after the explosion completes, should re-appear. Use OO design.
4. NPC explosions: use chunks and/or frames.
5. Demonstrate object pooling and show pool contents in your information HUD that appears at program start, toggles with F1, and shows information about how to move the player, how many objects are active and how many objects are pooled, and any other information you wish to display.
6. Implement a reset function that restarts your game; toggle reset with "r" key.

Your assignment will be tested on a Linux platform using gcc or clang, however you should test your project on several different platforms and it should be independent of platform and language implementation. (Key summary: F1 ⇒ help, F4 ⇒ frames, and r ⇒ restart)