**Testing**

The way we test our feature is by using the in-game level editor to create a stage that includes player and our new feature. After our team makes sure the Gold Bomb functions, we will check if any other subsystems got affected after our modification of Gold Bomb.

First of all, we need to test if our new feature works between Gold Bomb and player. For example, we test if the projectile and Gold Bomb interacts as we expected. If Intense Gold Bomb is freezable, we will make player throws an iceball at Gold Bomb. The iceball must rebound and does not harm or freeze the Gold Bomb. If player throws a fireball at Gold Bomb, it should reacts as same as before, the Gold Bomb need to be harmed and does not rebound the fireball.

After checking both interactions with projectile, we will test the death effect after Gold Bomb got killed. Such as, the detonation size of the Gold Bomb must be larger than originally, player must have the fire flower power up after the Gold Bomb got killed, and if the player already has the fire flower power up, the player should keep the fire flower power up ability.

If Intense Gold Bomb functions as expected, then our team will check on subsystems that might be affected after our modification of the game. For example, gold bomb and normal bomb both uses explosion file, so we need to check if normal bomb’s detonation size changed as well. Also, we need to adjust sprite animation according the detonation size as well. After we found out what subsystems got affected, we will test if it’s necessary to modify our feature in a different way that will have minimal affect to other subsystems. If so, we will find other solution and go back to our first step and test its functional ability again.

After all these testing are done, we need to run through the whole game and make sure our new feature works every single stage. If at any stage there’s an unexpected error, we will compare between current stage and original stage to find out the problem and modify the codes. If there’s no unexpected error, then the testing is done.

**Closing**

Through the design and implementation of our Gold Bomb to become a more interesting and rewarding badguy, we learned many lessons around game architecture and design: SAAM analysis, discussion of alternatives, team issues, concurrencies, sequence diagrams, impacted architecture, lessons learned, limitations, and so much more. We are ultimately pleased with the enhancements of giving the Gold Bomb the ability to explode with more flair, give Tux a Fire Flower upon death, and be invincible against iceballs. Though to Tux the Gold Bomb may be a badguy, the Gold Bomb will forever be a “goodguy” in our hearts, having taught the TUXEDitOrs a taste of the difficulties and joys of game development. Sometimes the best games don’t need major enhancements, but small ones to really make them glow.