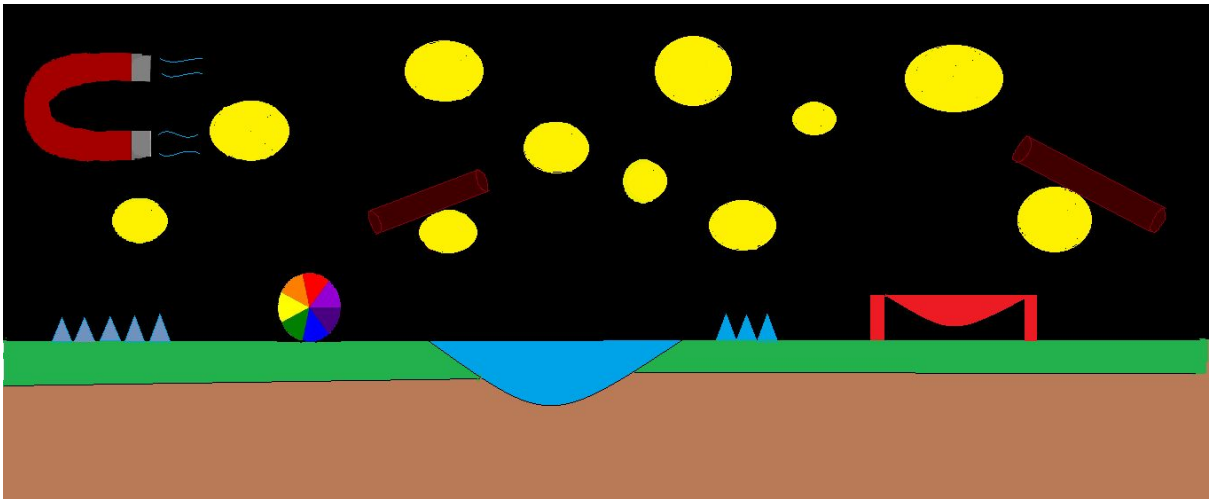
The Problem:



This will be an arcade game (inspired by http://www.pacman4u.com/pacman-killer/) where the player controls a metal player and uses it to land on and destroy other flying balls constantly traversing the screen from left to right. On landing on the flying balls, they disappear and the player gets launched upwards a little higher (Like Mario jumping on enemies).
Define your own rules for the points given for each destroyed ball. The player will level up when certain scores are attained and that will cause some additional features to get unlocked increasing the difficulty of the game. The primary objective is to score as much as possible.
In the following sections, the minimum requirements are mentioned. Use your imagination and enhance the game as per your liking. It is your game.
You should provide a single page quick start guide to those who want to play the game (aka TAs), describing the additional controls (basic controls should be as described below), and additional features.
Keep track of the scores and levels. They can be displayed on the screen too for bonus marks.

The World:

The world consists of a floor with a small semicircular pond and a trampoline. Flying balls of different colours (each corresponding to certain points) will traverse the screen above the ground at various heights (random) from left to right. The balls need to be densely located so

that its possible to destroy multiple balls using the boost obtained by destroying one. Some of the flying balls should have a rectangular slab attached at different angles, if the players lands on these slopes he/she would be launched at an angle of reflection from the perpendicular to the slab following the laws of reflection. Later, a magnet should appear and disappear randomly on either sides of the game which will cause the player's motion to be influenced (make sure projectile like motion [constant horizontal acceleration] is followed). At a later level, spiked porcupines should appear which if the player lands on will result in a penalty incurred and the porcupine disappears. Your initial game can have fixed boundaries where attempting to cross the boundaries would not work and you would be reflected back. For bonus, you can create a game without boundaries where the world can be extended indefinitely (similar to a Mario game).

Objects and Physics:

You should incorporate the basic laws of motion (Newton's Laws) into the behaviour/ motion of the objects. The minimum set of factors you need to consider are:

1. Water: When the player is inside the water body, movement speed is slowed down and the height reached by jumping from inside is also shorter. The floor of the water should be a semi-circle and the player on falling in the water shouldn't bounce off, but follow the curve until the lowermost point is reached (provided the player isn't explicitly moving it sideways). If the player is moving left or right, he should follow the path along the semi-circle.
2. Inclined slopes on some flying balls: If the player lands on these slopes, he/she would be launched at an angle of reflection from the perpendicular to the slab following the laws of reflection.
3. Trampoline: Landing on the trampoline will cause the player to jump higher than usual (Make sure the player starts off faster and eventually slows down and doesn't just reach a higher point with the same speed).
4. Magnet: The player's path needs to be influenced by the magnet. Assume magnet causes a constant attractive force in its direction.

Controls:

The player is controlled by the arrow keys: A to go left, D to go right and SPACE to jump. The player can also be made to move left or right by dragging on the screen. Make sure multiple keys can be pressed at the same time.
To control the camera: scroll wheel to zoom in/ out and arrow keys to pan.

Hints:

1. When you create objects in the world, make them as objects in the program (class, struct) with all the parameters needed for drawing, animating, motion, sound etc. built into the object along with a method to draw themselves. These parameters would include, but not limited to position, orientation, color, state, etc. This way, you will be

able to replicate an object easily and enhance them later on.

2. The main control logic (or game engine) will look at the current state (time, collisions, etc.) and update each element in the world. Create a collision detection function that checks for collision between any two objects. This may be used by the game engine to find any impact and take corresponding action.

3. Start with a simple world and complete the game. You may enhance the objects/ motion, once you are done with your V1.0. While creating the objects and the game engine, think of how to make each parameter that you want to be flexible.

Version 1.0: Flying balls, floor, player and slopes on some flying balls. Basic physics
Version 2.0: Water, Trampoline and accompanied physics. Zooming and panning.
Version 3.0: Moving porcupines, Magnet and accompanied physics.
Version 4.0: Use your imagination! :)