**Plan de Solución para el Parcial III**

* Se utilizará interfaz grafica
* Se utilizarán clases para cada objeto en pantalla
  + Una clase cañón
    - Atributos
      * Float Posición en X
      * Float Posición en Y
      * Int Identificador, para variar entre cañón defensivo y ofensivo
    - Métodos
      * Constructor de clase (float posX, float posY)
      * BoundingRect
      * Painter
  + Rango\_accion
    - Atributos
      * Float Posición en X
      * Float Posición en Y
      * Int identificador, para variar entre cañón ofensivo y defensivo
      * Float Rango
    - Métodos
      * Constructor (float posX, float posY)
      * BoundingRect
      * Painter
  + Proyectil\_Fisicas
    - Atributos
      * Float Posición en X
      * Float Posición en Y
      * Float Ángulo
      * Float Velocidad inicial
      * Float Velocidad en X
      * Float Velocidad en Y
      * Float Radio
      * Float Aceleración en X
      * Float Aceleración en Y
      * Float Delta de tiempo
      * Float Rango de daño
      * Int Identificador, para variar entre proyectil ofensivo y defensivo
    - Métodos
      * Constructor (float Px, float Py, float Angulo, float velocidad inicial, float radio)
      * Destructor
      * Actualizar Posición
      * Calcular velocidad
      * Métodos get y set necesarios
  + Proyectil\_Graph
    - Atributos
      * Instancia de la clase Proyectil\_fisicas
      * Int Id, para variar entre proyectil ofensivo o defensivo
    - Métodos
      * Constructor (float Px, float Py, float Vel\_In, float angle, float range)
      * Destructor
      * BoundingRect
      * Painter
      * Actualizar (float Limite\_inferior)
      * Métodos get y set necesarios
* Los elementos se presentarán al usuario a través de un QGraphicsView y los parámetros de la simulación se ingresan a través de varios DoubleSpinBox.
* Por medio de pushBottons se solicitará al usuario la simulación que desea efectuar.
* Se tendrá en cuenta los tiempos de reacción despectivos para cada cañón.