Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Proyecto
- Fase 2: Simulación Completa del Juego de Pool

En esta segunda fase del proyecto, se agregarán un conjunto de nuevas funcionalidades al proyecto 1 con la finalidad de completar el juego de pool. Dichas funciones a integrar son las siguientes:

- Agregar la interacción con las paredes de la mesa en la fase de simulación
- Agregar la interacción con las troneras de la mesa en la fase de simulación
- Incrementar el número de esferas que se pueden cargar en la mesa
- Guardar y cargar el estado de la mesa

Aparte de estas funcionalidades nuevas, se desea que usted realice el proyecto haciendo uso del paradigma de Programación Orientado a Objeto.

1 Interacción con las paredes

La interacción con las paredes se puede dividir en dos fases, en la primera fase se detecta la posible colisión con alguna de las paredes y en la segunda fase se debe solventar la misma.

1.1 Detectar la colisión

Detectar la colisión entre una pared de la mesa y una de las bolas es una tarea simple ya que las paredes están alineadas a los ejes; para saber sí una bola no está colisionando con una pared solo hay que comprobar si la misma se encuentra totalmente contenida dentro de la mesa, si parte de ella supera el límite impuesto por una pared o alcanza el límite, entonces la bola está chocando con la pared.

Otro punto que se debe tomar en cuenta al momento de calcular la colisión con una pared es si la velocidad que lleva la misma traerá como consecuencia que se separe de la pared con la que está colisionando, lo que indicaría que la colisión ya fue solventada en una iteración previa.

Es importante que no solo se detecte si la bola está colisionando con las paredes de la mesa, sino que se identifique con cual pared lo hace, ya que es necesario conocerlo para solventar la colisión.

En la Figura 1 se puede observar un ejemplo en el cual las bolas rojas representan las bolas en la que se detectó colisión con la pared.

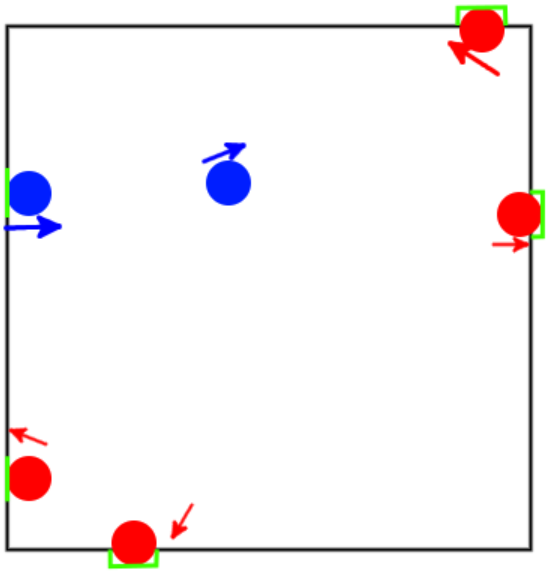


Figura 1: Las bolas de color rojo son aquellas que están colisionando, las azules las que no se encuentran colisionando. Las flechas indican la velocidad/dirección de cada bola y las líneas verdes la separación que tiene la bola de la pared con la que colisiona.

1.2 Solventar la colisión

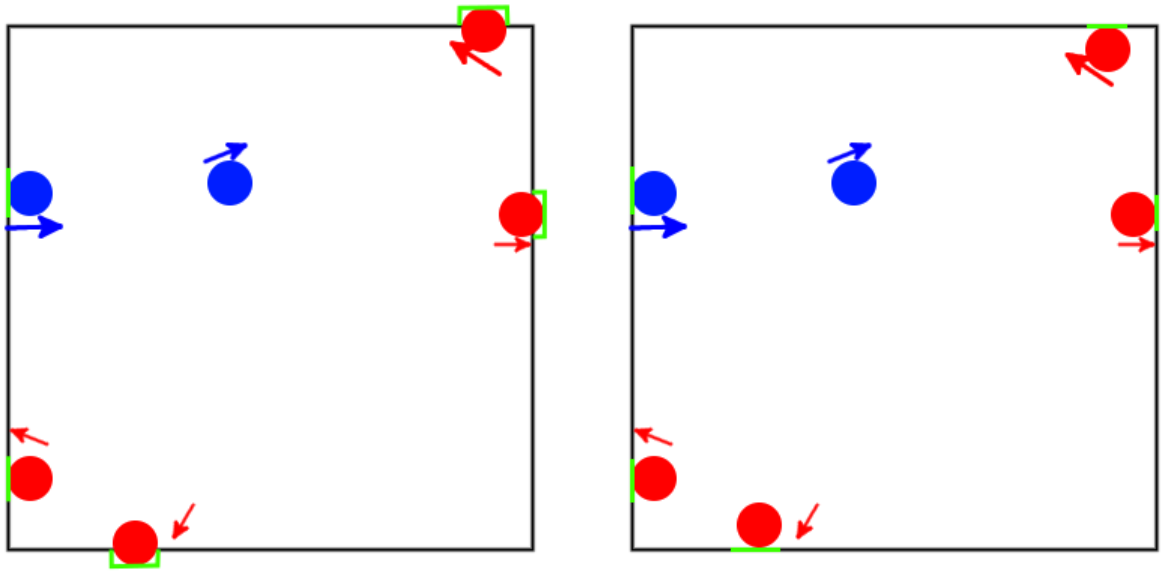
Si en la fase anteriormente descrita se obtuvo como resultado que existe colisión con una pared de la mesa se deben realizar dos acciones.

Primero se debe separar la bola de la pared con la que colisiona lo mínimo posible para que la misma solo toque el borde de la mesa (véase Figura 2).

Luego se debe calcular la nueva velocidad que tendrá la bola como resultado de la colisión, para calcular dicha velocidad se debe evaluar si la pared con la que colisiona es horizontal o vertical.

- Si la pared es vertical, la nueva velocidad es la misma que ya poseía, pero se invierte el signo en la componente X
- Si la pared es horizontal, la nueva velocidad es la misma que ya poseía, pero se invierte el signo en la componente Y

Como ejemplo gráfico puede observar la Figura 3, donde la mesa se encuentra representada como un cuadrado solo para propósitos demostrativos.



(a) Estado inicial (b) Estado después de la separación de las paredes

Figura 2: En la imagen (a) el estado inicial de la mesa en la iteración, con las velocidades asociadas. En la imagen (b) se puede apreciar las nuevas posiciones de las esferas luego del proceso de separación.

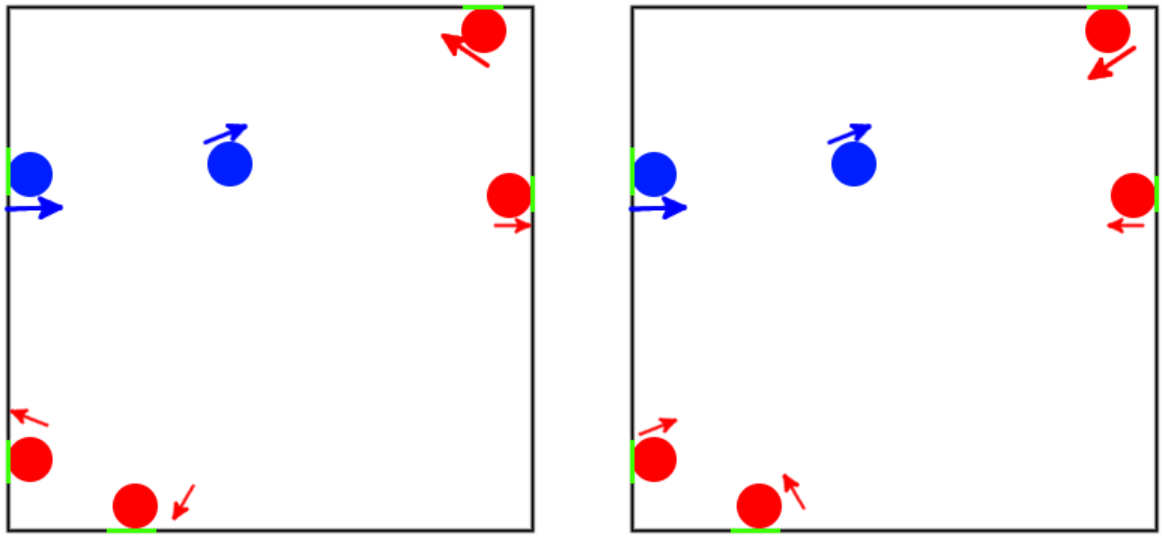
2 Interacción con las troneras

Para saber si una bola se encuentra dentro de una de las seis troneras de la mesa se debe de evaluar si el círculo que describe la tronera se intercepta con el círculo que describe la bola, haciendo uso de la fórmula de intersección círculo - círculo dada en el proyecto 1.

En caso de que la bola intercepte la tronera se debe detener la misma en el punto en el que se encuentra y debe ser ignorada durante el resto de la simulación, ya que la misma entró en una tronera y ya no se puede interactuar con ella.

3 Mayor cantidad de bolas en la mesa

En esta fase del proyecto debe hacer uso de estructuras que le permitan cargar en la mesa hasta un máximo de 16 bolas, incluyendo la bola blanca.



(a) Estado inicial

(b) Estado después de la fase de cambio de velocidades

Figura 3: En la imagen (a) la configuración luego de la separación de los bordes, con las velocidades asociadas. En la imagen (b) se puede apreciar el cambio de las velocidades en las bolas que se encontraban colisionando.

4 Guardar y Cargar el estado de la mesa

Debe agregar una opción en el menú "Jugar", que le permita guardar la configuración de la mesa y el estado actual de las bolas en la misma, en un archivo de texto. A su vez debe poder realizar la opción inversa y poder cargar la configuración de una mesa con sus bolas desde un archivo de texto.

En el mismo debe guardar toda la información necesaria para representar el último estado del juego, lo cual corresponde a:

- Mínimo y máximo de la mesa
- Posición de cada bola
- Número de cada bola
- Si la bola se encuentra dentro de una tronera o no
- En caso de encontrarse dentro de una tronera, en que tronera se encuentra (ver proyecto 1)

5 Resultados de una simulación

Una simulación debe arrojar como resultado en esta fase los siguientes puntos:

- La posición final de cada bola
- El número que identifica a la bola
- Si la misma se encuentra en una tronera, indicando cual, o si no se encuentra

6 Entregables

Para esta fase se van a realizar dos entregas. En la primera entrega se deberá generar un **Informe de Análisis y Diseño** que debe contener los siguientes puntos:

- Fórmulas y procedimientos para manejar la colisión con las paredes de la mesa
- Estructura de datos para permitir agregar más bolas en el juego y explicación de la forma en que se utilizará la misma
- Estructura para el archivo de texto en el que se guardará el estado de la mesa
- Una representación gráfica de las clases que se planean crear, con los atributos, métodos y modos de acceso, según el paradigma Programación Orientado a Objeto
- Explicación del diagrama de clases solicitado en el punto anterior

El segundo entregable será la **Aplicación Desarrollada** luego de tomar en consideración las correcciones que se realicen al diseño propuesto en el informe.

7 Reestricciones

- El proyecto debe ser desarrollado en C++. Sugerimos la utilización de geany como ambiente de desarrollo en Linux y en Windows sugerimos el uso de Dev C++
- Cualquier copia será penalizada para todos los involucrados con la nota de CERO (0) puntos, además del reporte respectivo al Profesor de la Sección y a la Coordinadora de la Materia
- Todo código enviado debe compilar correctamente
- Puede hacer uso de todo el contenido visto en la materia

8 Consultas

- En caso de dudas, consultar al Profesor Fracisco Moreno, francisco.morenociens@gmail.com
- En el sitio web de la materia se publicarán aclaratorias y respuestas a preguntas frecuentes sobre el proyecto

9 Entrega

9.1 Entrega del Informe

- El **Informe de Análisis y Diseño** debe ser realizado de manera individual y su fecha de entrega será hasta el Viernes 5 de Febrero a las 11:59 pm.
- El mismo debe ser enviado por correo electrónico al Prof. Francisco Moreno, francisco.morenociens@gmail.com
- El asunto del correo debe cumplir el siguiente formato: AyP-Informe-Proyecto2-Sección-Cédula.
- En el mismo solo debe estar adjunto el informe en formato .pdf, el cual debe estar nombrado bajo el siguiente formato Informe-Proyecto2-Seccion-NombreyApellido-Cedula.pdf, sin acentos o ñ.

9.2 Entrega de la Aplicación

- El proyecto debe ser realizado de manera individual y su fecha de entrega será hasta el Viernes 26 de Febrero a las 11:59 pm.
- El mismo debe ser enviado por correo electrónico al Prof. Francisco Moreno, francisco.morenociens@gmail.com
- El asunto del correo debe cumplir el siguiente formato: AyP-Proyecto2-Sección-Cédula.
- En el mismo solo debe estar adjunto el archivo .cpp, el cual debe estar nombrado bajo el siguiente formato Proyecto2-Seccion-NombreyApellido-Cedula.cpp, sin acentos o ñ.