



Universidad de
Talca
Sede Curicó

Computación gráfica

Tarea 2

Alumno	Erwin Ried
Profesor	Francisco Reyes
Fecha	20 de noviembre de 2007

CONTENIDO

Contenido.....	2
Tabla de ilustraciones.....	3
Introducción.....	4
Análisis del problema.....	5
Diseño.....	6
Creación de los pinos.....	7
Creación de la Bola.....	11
Creación de la pista de juego.....	12
El lanzamiento de la bola.....	13
Colisión con los pinos.....	19
Las texturas y el entorno.....	21
Conclusiones y comentarios finales.....	24

TABLA DE ILUSTRACIONES

Ilustración 1: Marcador de ejemplo de bowling.....	5
Ilustración 2: Medidas usadas para el pino	7
Ilustración 3: Silueta del pino, ajustada y simétrica	7
Ilustración 4: Puntos ajustados al contorno de la silueta	8
Ilustración 5: Coordenadas del contorno	8
Ilustración 6: Contorno de la mitad, ajustado inicialmente a un polinomio.....	9
Ilustración 7: Curva cúbica de Bezier	9
Ilustración 8: Gráfico de las tres curvas del contorno	10
Ilustración 9: Bola de Bowling.....	11
Ilustración 10: Pista de Bowling.....	12
Ilustración 11: Contorno de la pista.....	12
Ilustración 12: Efectos y trayectorias de la bola	13
Ilustración 13: Límites de pista para la trayectoria del tiro.....	14
Ilustración 14: Colisión de la bola con un pino	19
Ilustración 15: Posiciones de los pinos	20
Ilustración 16: Colisión base	20
Ilustración 17: Imágenes y texturas utilizadas.....	21
Ilustración 18: Foto original de la pista	22
Ilustración 19: Segmento de pista	22
Ilustración 20: Textura en su parte final.....	22
Ilustración 21: Usando el filtro de "Offset"	23
Ilustración 22: Usando la textura final.....	23

INTRODUCCIÓN

El bolo americano (bowling, conocido como boliche en México) es un deporte que se juega en recintos cerrados, consistente en derribar un conjunto de piezas de madera (llamados bolos o pines) mediante el lanzamiento de una pesada bola contra ellos.¹

En esta tarea, la idea es crear un juego de Bowling 3d usando los conocimientos adquiridos en el transcurso del curso de computación gráfica. El bowling es un deporte de precisión que de forma similar al billar suele ser de una precisión matemática.

Algo interesante que trataré de conservar en el transcurso de la tarea será mantener las medidas reales en lo posible. Una unidad en OpenGL será entonces un centímetro de la vida real, y así toda la escena y sus componentes serán diseñados a escala real. Siguiendo con esta esencia de realidad, se intentará realizar todo por medio de ecuaciones de descripciones absolutas e ideales, como por ejemplo curvas de contorno precisas para los pines y el canal de la pista, ecuación del lanzamiento aplicando efectos (spin), fuerza de tiro y otras características.

Un punto destacable es que en el código y la interfaz de juego intentaré utilizar inglés como idioma base, por lo que los objetos mantendrán nombres como Bowling, Bowling Pin y otros similares con el fin de evitar equivocaciones implícitas en los nombres no anglosajones.

¹ http://es.wikipedia.org/wiki/Bolo_americano

ANÁLISIS DEL PROBLEMA

Hay una serie de dificultades presentadas por este trabajo, en particular:

- ❖ La creación de una escena
 - ❖ Pinos, bola, pista
 - ❖ Salón
- ❖ La simulación del tiro de la bola
- ❖ La colisión con los pinos

Adicionalmente podría considerarse la conjunción de todas estas características a modo de juego (marcador, opciones de tiro, entre otros) como una dificultad adicional.

Se debe tomar especial atención al sistema de puntuación utilizada en el bowling:

Por cada bolo derribado, se sumará un punto al marcador del jugador, siempre y cuando no se den las siguientes circunstancias:

Haber derribado todos los bolos en el primer lanzamiento de un juego (a esta jugada se le suele llamar strike, pleno o chuza, y se representa en el marcador con una X). En este caso el jugador no podrá utilizar su segundo lanzamiento del juego. Se sumarán diez puntos, más los puntos totales que se consigan en los dos siguientes lanzamientos de bola.

Haber derribado todos los bolos utilizando los dos lanzamientos del juego (esta jugada se llama spare, semipleno o mediachuza, se representa en el marcador con un /). Se sumarán diez puntos, más los puntos que se consigan en el siguiente lanzamiento de bola.

La máxima puntuación posible es 300 puntos, necesitándose para ello conseguir 12 strikes consecutivos (de la tirada 1 a la 10 y sus dos adicionales correspondientes)²

Así el marcador del juego debe tener once recuadros, los primeros diez separados en tres sub recuadros, dos superiores y uno inferior y el onceavo sólo con dos sub recuadros, como muestra la siguiente imagen:



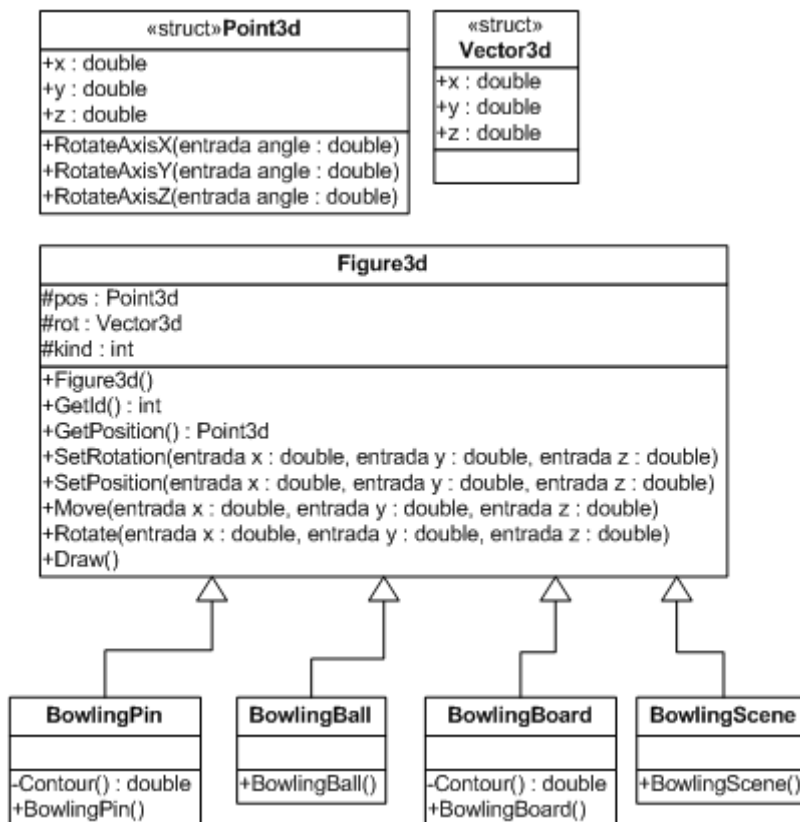
Ilustración 1: Marcador de ejemplo de bowling

² http://es.wikipedia.org/wiki/Bolo_americano#Sistema_de_puntuaci.C3.B3n

DISEÑO

Aunque en general el problema se ve bastante largo de abordar, una buena separación de componentes podría simplificar el diseño. En particular, existirá una clase heredable que definirá cada una de las figuras, así, se usará un identificador genérico para elementos estáticos, otro genérico para elementos estáticos de baja resolución y otros identificadores dinámicos para los objetos de juego.

Se puede obtener un diagrama de clases bastante sólido, en donde `Figure3d` exige la implementación de métodos virtuales como `Draw`. De esta forma la consistencia de los objetos tridimensionales es ideal, un tablero de juego, la bola, los pinos o cualquier otro objeto puede ser manejado por medio de un índice, por los mismos mecanismos que cualquier otro lo que nos simplifica el diseño:



En base, las figuras serán hechas por medio de ecuaciones para poder mantener una integridad visual ante diferentes situaciones. Se cuenta con la capacidad de poder generar objetos de baja definición para acelerar la visualización o de alta definición para cuando exista un acercamiento (por ejemplo a los pinos de la pista del usuario). Por medio de un parámetro podríamos generar un pino de baja resolución fácilmente:

```
Figure3d *figura = new BowlingPin([param1, param2, ..., paramN],
                                   BOWLING_LO_FI_SCENERY);
```

Y trabajar con esa figura como normalmente lo haríamos.

CREACIÓN DE LOS PINOS

Como son múltiples pinos simultáneos, el proceso de diseño del mismo se hace sólo una vez. Primero hay que considerar las medidas estándar de un pino:

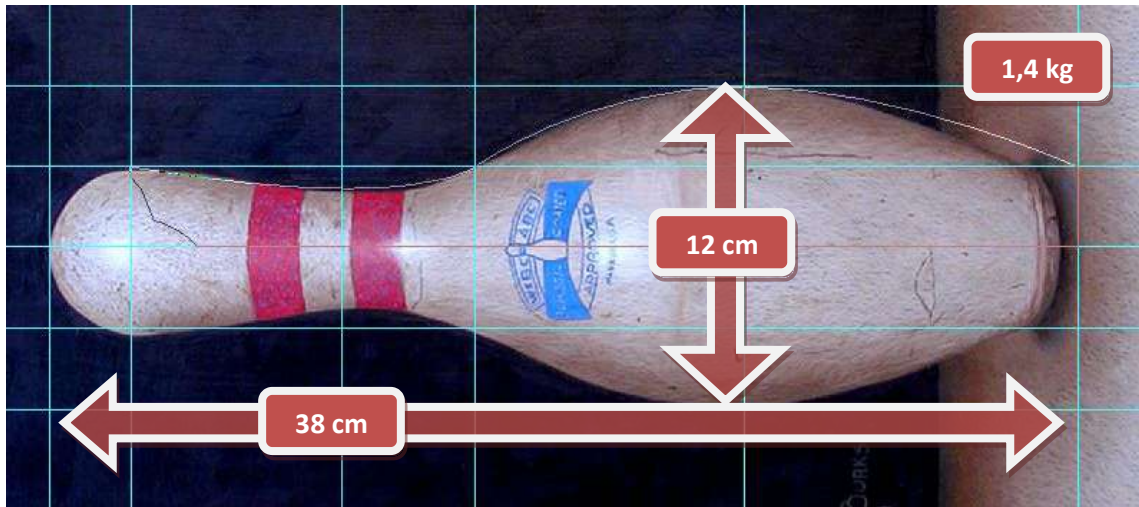


Ilustración 2: Medidas usadas para el pino

Ahora tomando en cuenta esas proporciones, es necesario buscar una imagen de un pino real. Esta imagen también servirá en su momento para generar una textura.

Con la imagen ajustada a un lienzo que tiene las medidas precisas de un pino, dibujo el contorno de la imagen, retocando las posibles fallas agregadas en la perspectiva del pino de la fotografía:

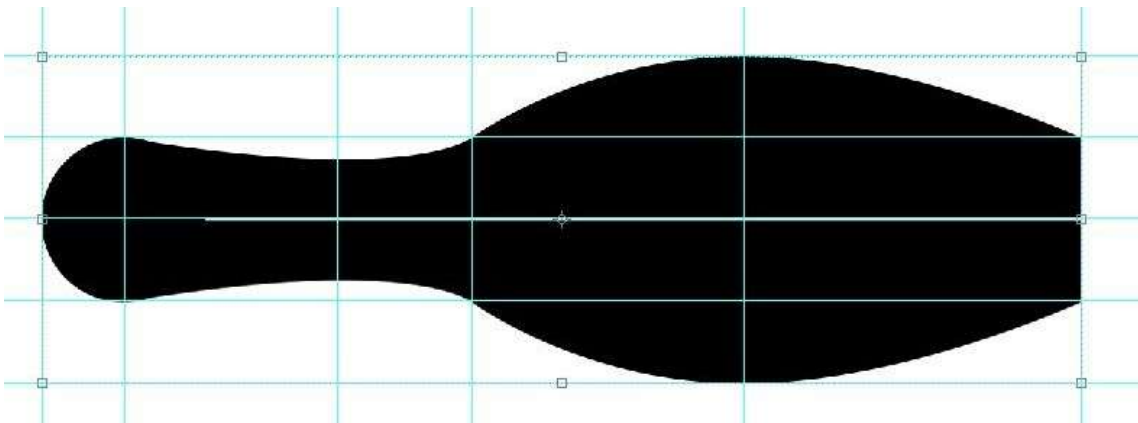


Ilustración 3: Silueta del pino, ajustada y simétrica

Usando la silueta y un programa de captura de pantalla adecuadamente configurado para reconocer los caracteres de la interfaz del programa de edición gráfica, capturo las coordenadas de todo el contorno a intervalos regulares, intentando obviar capturas en los puntos planos:

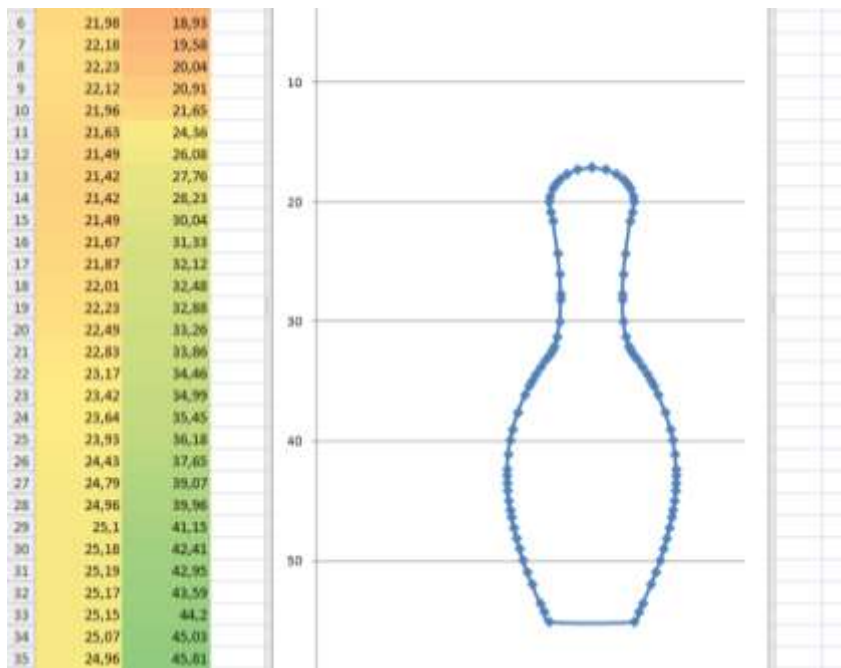


Ilustración 4: Puntos ajustados al contorno de la silueta

Así, obtengo un archivo de texto con todas las coordenadas del contorno de la silueta. De todas formas, aún es necesario ajustar los datos lo que se hace relativamente sencillo en una planilla de cálculo cualquiera.

Utilizando una interpolación podríamos encontrar una ecuación que se ajuste con precisión a todos los puntos del contorno. Una forma más sencilla es buscando un polinomio de un grado bastante alto para lograr el mismo efecto, aligerando los cálculos. Para esto, invertimos los puntos (de tal forma de lograr una curva horizontal) y nos preocupamos del lado superior de la ecuación como se ve en la Ilustración 6.

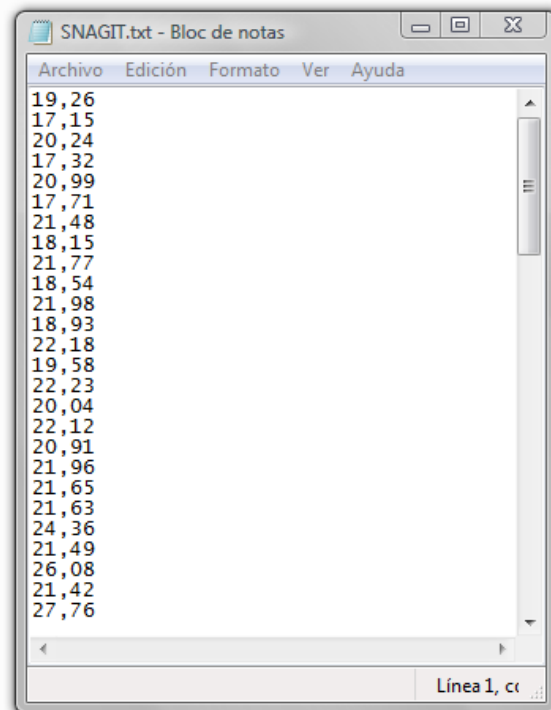


Ilustración 5: Coordenadas del contorno

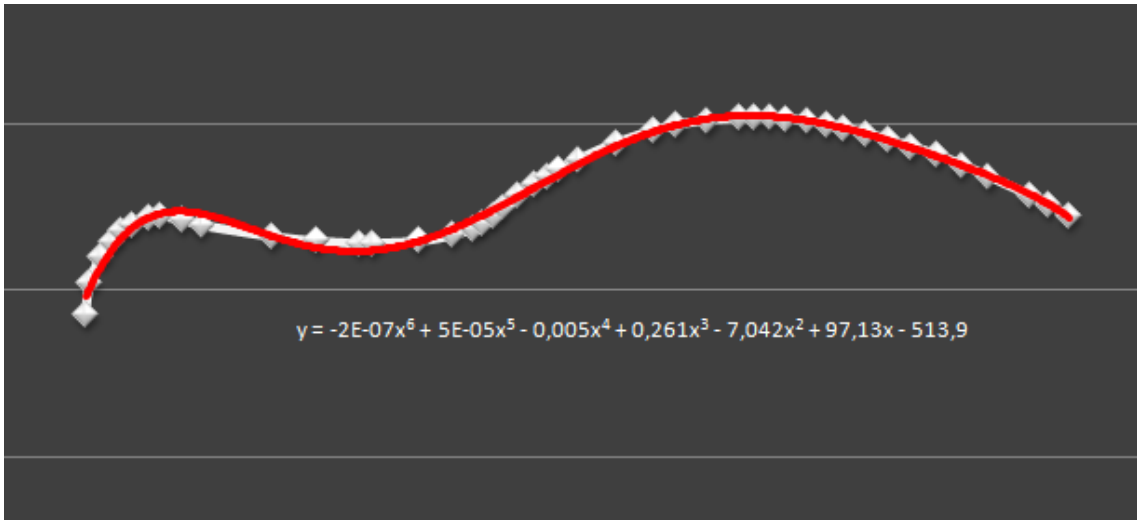


Ilustración 6: Contorno de la mitad, ajustado inicialmente a un polinomio

Sin embargo la interpolación entre estos puntos es de un alto grado de complejidad. Incluso la ecuación resultante de forma posterior podría no ser tan precisa como se espera debido a la extensión.

Para simplificar la tarea de la búsqueda del contorno, usare splines³, definiendo segmentos del contorno con ecuaciones sencillas. La verdad es que haré una especie de conjunción de curvas bezier⁴ por medio de dos puntos extremos, en donde los nodos modificadores de cada curva de extremo será definido por un punto en común, es decir de una tradicional curva de bezier cúbica como la mostrada en la Ilustración 7 uniré P_1 y P_2 calculando el polinomio por medio de una interpolación de Lagrange.

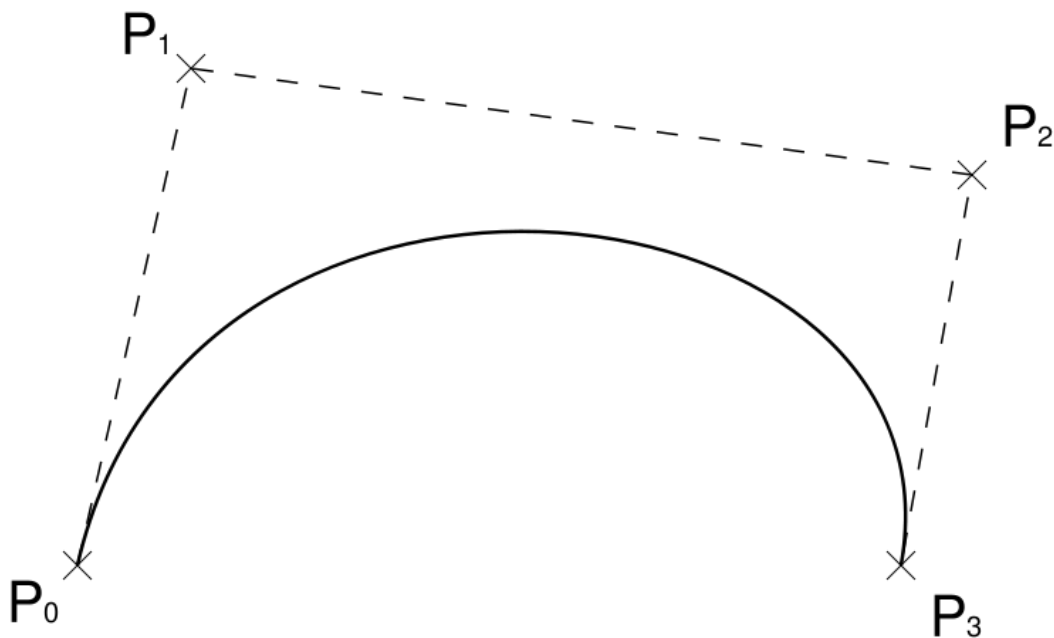


Ilustración 7: Curva cúbica de Bezier

³ <http://es.wikipedia.org/wiki/Spline>

⁴ http://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier

De esta forma nuestro contorno inicial se disgrega en un conjunto de curvas sencillas y acotadas, en efecto podemos notar algo del contorno al graficarlas las tres juntas:

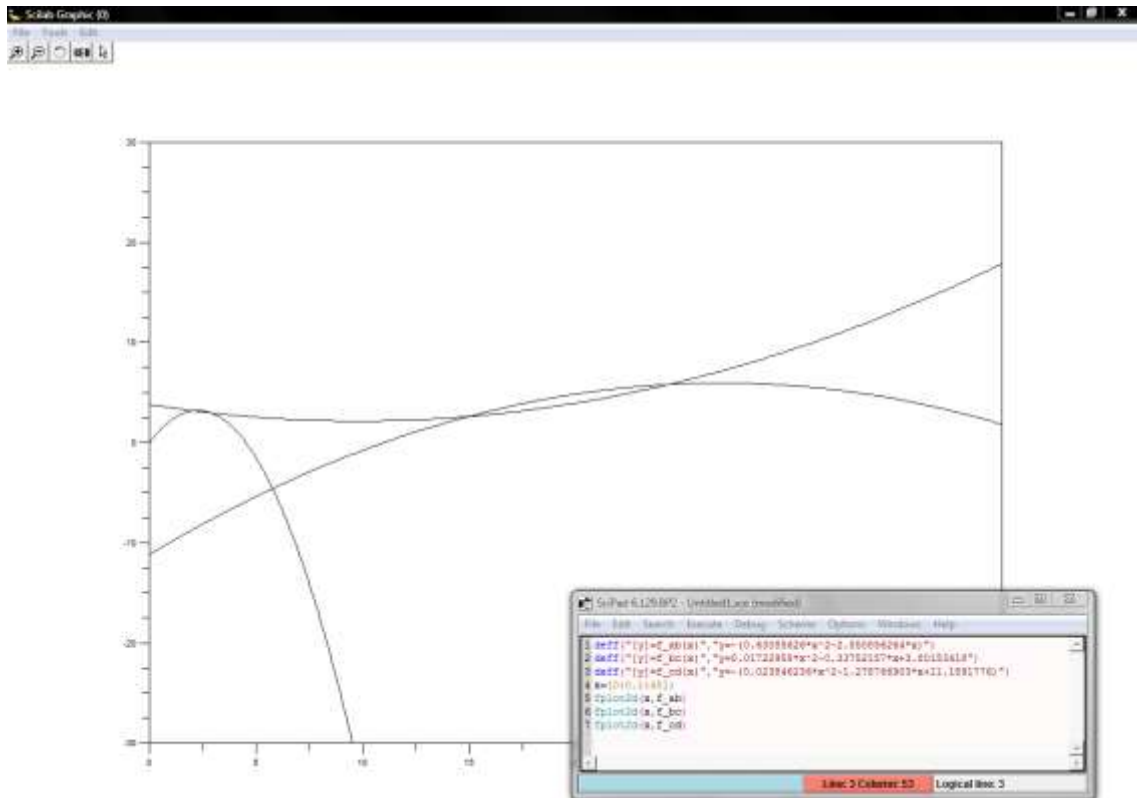


Ilustración 8: Gráfico de las tres curvas del contorno

Los puntos de las curvas mostrados en las imágenes nos sirven directamente para calcular las ecuaciones que quedan de la siguiente forma (en orden de izquierda a derecha, en el contorno):

$$f_{AB}(x) = -(0.63085626x^2 - 2.850856264x)$$

$$f_{BC}(x) = 0.01722958x^2 - 0.33752157x + 3.80153418$$

$$f_{CD}(x) = -(0.023846236x^2 - 1.278766903x + 11.1891776)$$

Claramente en el programa, el contorno se define por una función que cohesionara las tres ecuaciones anteriores por medio de condiciones (en donde la solución de la incógnita es la misma) y retornando cero para valores fuera del alcance⁵ del pino real.

Ahora sólo queda utilizar este contorno y hacer un sólido de revolución. El secreto ahora es rotar los puntos y como si fuera una suma de Riemann, usar sólo trozos del contorno. La rotación debe realizarse manualmente, aprovechando los métodos de la estructura Point3d que permiten rotar un punto (no se puede ir creando una textura usando la rotación de OpenGL directamente, pues no se puede modificar la matriz con `glRotated` entre `glBegin` y `glEnd`).

⁵ Las dimensiones de los puntos de contorno del pino originalmente se basan en un pino oficial estándar (en centímetros), pero las medidas pueden haberse visto alteradas en el proceso de aproximaciones a curvas.

CREACIÓN DE LA BOLA

En particular, la bola es sencilla. Las medidas para ella serán:

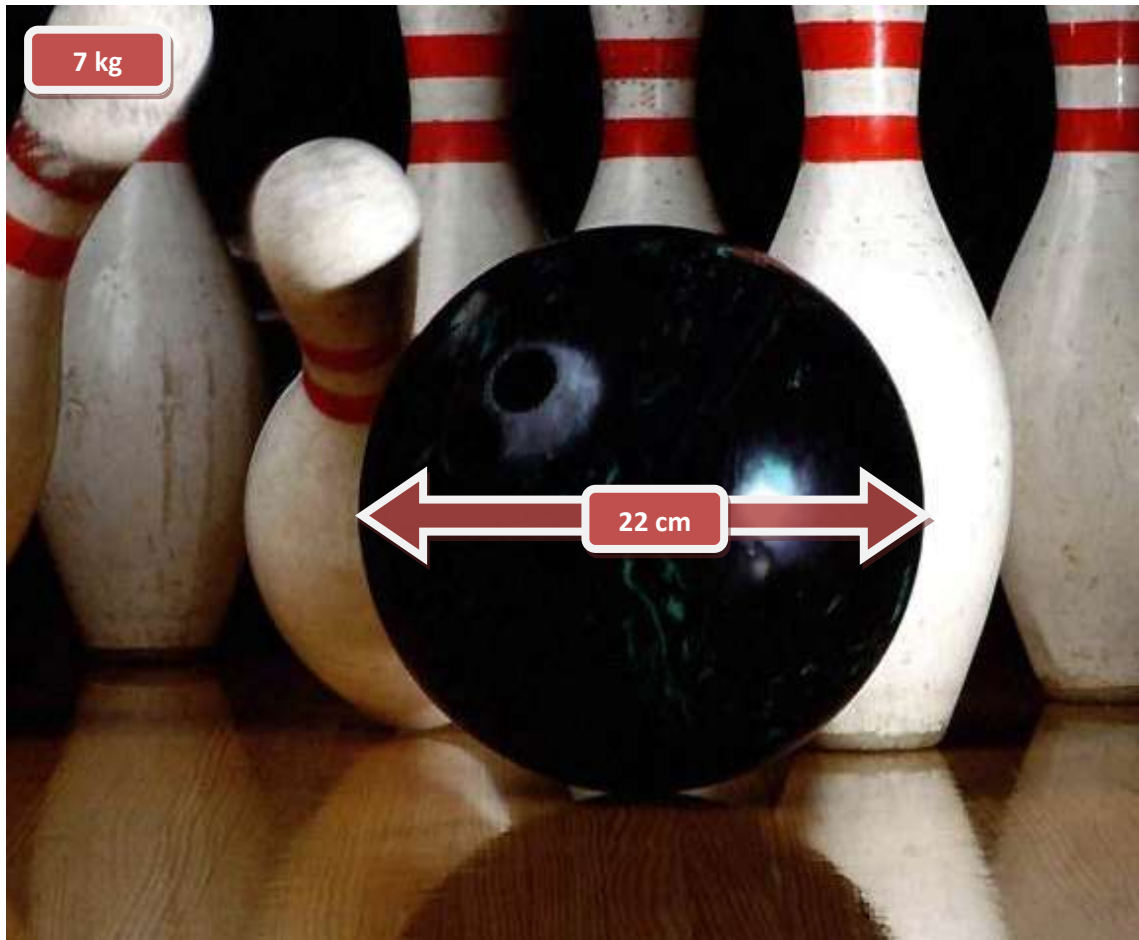


Ilustración 9: Bola de Bowling

La bola se crea simplemente con `glutSolidSphere` de la librería GLUT.

Aunque se creó una textura para la bola, al aplicarla por medio de la función antes citada, al no tener un control directo de cómo se aplica la textura misma, no se consigue un efecto óptimo por lo que finalmente no se utilizó ninguna textura.

CREACIÓN DE LA PISTA DE JUEGO

De forma similar a la del pino, la base para crear la pista de juego es un contorno especial, en esta ocasión de forma plana y no como sólido de revolución. Siguiendo unas medidas adecuadas tenemos que el tablero es así:



Ilustración 10: Pista de Bowling

Los laterales miden algunos centímetros más que la bola. En esta situación el contorno nos queda así:

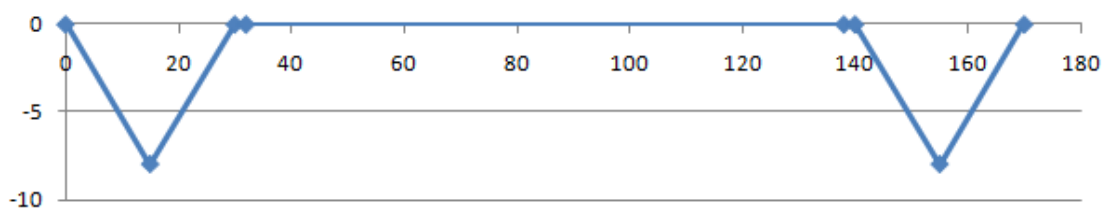


Ilustración 11: Contorno de la pista

Con estos datos es fácil encontrar las dos formulas que generarán los contornos laterales para la pista:

$$f_1(x) = 0.03556x^2 - 1.066666x$$

$$f_2(x) = 0.03556x^2 - 11.02222x + 846.22222$$

EL LANZAMIENTO DE LA BOLA

En el bowling, el tiro tiene una particularidad. Debido a que el roce de la bola es relativamente despreciable al inicio de la pista (principalmente por la fuerza que lleva y el aceite de la pista) se produce un efecto casi siempre en el tramo final del trayecto.

Así entonces tenemos un tiro según el efecto aplicado a la bola, desde una vista superior, con el final de la pista en la zona del punto uno:

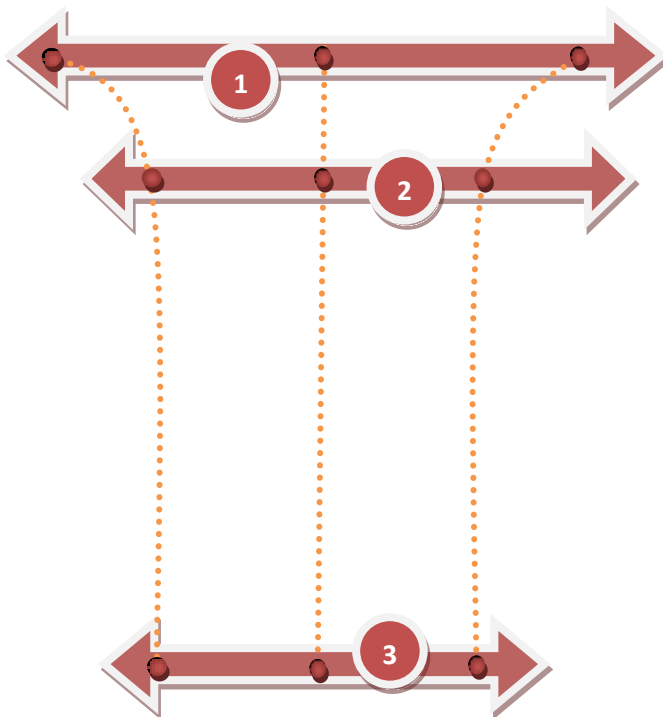


Ilustración 12: Efectos y trayectorias de la bola

Así, podemos disgregar las características del tiro en una serie de parámetros que alterarán esas trayectorias. Para aclarar las cosas se definirán intervalos válidos de las variables:

- ❖ Fuerza del tiro: Modificará la velocidad inicial a la que se le irá aplicando una fricción constante. Modifica la altura del punto dos de la trayectoria.

$$0 \leq F \leq 10$$

- ❖ Efecto del tiro: Modifica la posición horizontal del punto uno en la curva de la trayectoria.

$$-45 \leq E \leq 45$$

- ❖ Posición inicial del tiro: Modifica la posición de los tres nodos de la trayectoria. Hay que considerar que la pista tiene 106 cms de ancho, en este caso la variable representa la posición desde el centro.

$$-50 \leq P \leq 50$$

- ❖ Dirección inicial del tiro: Modifica la posición del nodo uno y dos de la trayectoria, También, hay una relación directamente proporcional entre las dos posiciones nuevas de los nodos.

$$-15 \leq D \leq 15$$

Luego de tener todos estos valores se puede confeccionar una interpolación de la trayectoria. La idea es mantener la simulación dentro de los parámetros de la pista:

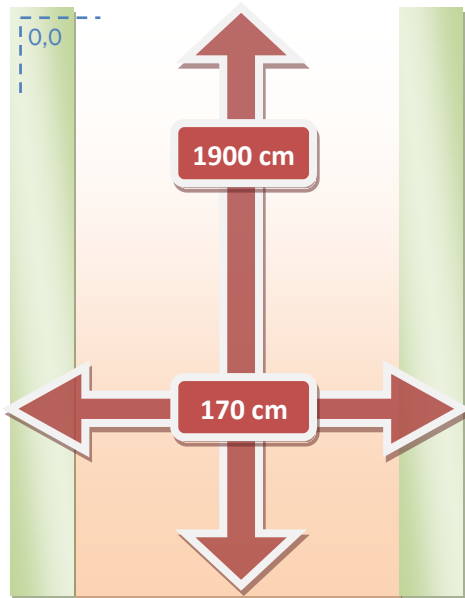


Ilustración 13: Límites de pista para la trayectoria del tiro

F E P D

Los tres puntos del tiro tendrán posiciones definidas por los parámetros considerados anteriormente. Primero se deben centrar:

$$P1_x(P) = (P + 70)$$

$$P2_x(P) = (P + 70); P2_y(P) = 700$$

$$P3_x(P) = (P + 70)$$

A continuación se aplica la conversión según la fuerza del tiro:

$$P1_x(P, F) = (P + 70)$$

$$P2_x(P, F) = (P + 70); P2_y(P, F) = 700 - (50 * F)$$

$$P3_x(P, F) = (P + 70)$$

Según la dirección inicial sólo se modifican los dos primeros puntos, la relación entre ellos será de 1:4, para evitar una inclinación excesiva:

$$P1_x(P, F, D) = (P + 70) + D$$

$$P2_x(P, F, D) = (P + 70) + \frac{D}{4}; P2_y(P, F, D) = 700 - (50 * F)$$

$$P3_x(P, F, D) = (P + 70)$$

Finalmente con el efecto aplicado:

$$P1_x(P, F, D, E) = (P + 70) + D + E$$

$$P2_x(P, F, D, E) = (P + 70) + \frac{D}{4}; P2_y(P, F, D, E) = 700 - (50 * F)$$

$$P3_x(P, F, D, E) = (P + 70)$$

Los tres nodos del tiro nos quedan de la siguiente forma:

$$P1_x(P, D, E) = P + 70 + D + E; P1_y = 0$$

$$P2_x(P, D) = P + 70 + \frac{D}{4}; P2_y(F) = 700 - (50 * F)$$

$$P3_x(P) = (P + 70); P3_y = 1900$$

Con una función de interpolación en tres puntos, aplicando nuevamente Lagrange conseguimos la ecuación de la posición según los parámetros del tiro. En efecto:

$$T1_f = F + (17300 * P - (17300 * X - 1211000))$$

$$T1 = 5700 * D^3 + \left(13300 * E + \left((800 * P - (800 * X - 56000)) * T1_f\right)\right) * D^2$$

$$T2_f = \left(\left(1600 * P - (1600 * X - 112000)\right) * F + (38400 * P - (38400 * X - 2688000))\right)$$

$$T2 = 7600 * E^2 + T2_f * E$$

$$T3 = (800 * P^2 - (1600 * X - 112000) * P + (800 * X^2 - 112000 * X + 3920000)) * F$$

$$T4_f = (11600 * X^2 - 1624000 * X + 56840000)$$

$$T4 = (11600 * P^2 - (23200 * X - 1624000) * P + T4_f)$$

$$T5 = \left(\left(800 * P - (800 * X - 56000)\right) * F + (19200 * P - (19200 * X - 1344000))\right) * E^2$$

$$T6 = (800 * P^2 - (1600 * X - 112000) * P + (800 * X^2 - 112000 * X + 3920000)) * F$$

$$T7 = (19200 * X^2 - 2688000 * X + 94080000)$$

$$T7 = (19200 * P^2 - (38400 * X - 2688000) * P + T7_f)$$

Para tener finalmente nuestra función que nos dice la posición, en cualquier instante, según las características del tiro:

$$T(P, F, D, E, X) = \frac{T1 + (T2 + (T3 +)) * D + T4(T5 + (T6 + T7) * E)}{3 * D^3 + 7 * E * D^2 + 4 * E^2 * D}$$

Esto nos hace reflexionar sobre la complejidad que tiene un sencillo proceso como el lanzamiento de una bola. De hecho la ecuación a pesar de ser bastante larga, es simple y no involucra fuerzas gravitacionales, roces, índices por material ni nada por el estilo y sólo sería un tiro fidedigno en condiciones ideales.

Esta situación nos puede traer a la mente el hecho de que probablemente aún no sabemos si hay un dios que juega a los dados o no:

Einstein was very unhappy about this apparent randomness in nature. His views were summed up in his famous phrase, 'God does not play dice'. He seemed to have felt that the uncertainty was only provisional: but that there was an underlying reality, in which particles would have well defined positions and speeds, and would evolve according to deterministic laws, in the spirit of Laplace. This reality might be known to God, but the quantum nature of light would prevent us seeing it, except through a glass darkly⁶.

Probablemente sea inadecuado llevar a cabo los cálculos anteriores, dada su complejidad y peso de proceso. La solución en esta instancia es implementar un simplificado algoritmo sin perder el realismo de las características aplicadas al tiro, se usarán los mismos parámetros y límites anteriores:

❖ Fuerza del tiro:

$$0 \leq F \leq 10$$

❖ Efecto del tiro:

$$-45 \leq E \leq 45$$

❖ Posición inicial del tiro:

$$-50 \leq P \leq 50$$

❖ Dirección inicial del tiro:

$$-15 \leq D \leq 15$$

El formato reducido de la ecuación de tiro tiene condiciones sencillas. Primero que todo, la velocidad inicial del tiro está definida por una constante, la cual es reducida ligeramente en cada iteración del ciclo de actualización de juego (60 cuadros por segundo):

$$Velocidad_{Nueva} = Velocidad - 0.05$$

Esta fricción cinética simulada es atenuada por la fuerza de tiro, la cual modifica la velocidad de la bola en una manera extremadamente intuitiva, adjuntándose a la expresión anterior:

$$Velocidad_{Nueva} = Velocidad - [0.05 \cdot (Fuerza + 1)]$$

Ahora, el efecto del tiro (spin) sólo afecta el trayecto final de la bola. El impacto que produce sobre la trayectoria es una modificación de su posición en el plano horizontal que se ajusta a:

⁶ <http://www.hawking.org.uk/lectures/dice.html>

$$Posición_{Horizontal\ nueva} = Posición_{Horizontal} + \left[\frac{Efecto \cdot 40}{Posición_{Horizontal}} \right]$$

De esa manera, mientras más cerca de los pinos esté la bola en su trayectoria, más efecto tendrá, tal como se puede apreciar en tiros reales del juego.

La posición inicial del tiro es evidente, modifica sólo la posición horizontal en una instancia inicial y no vuelve a aplicarse.

La dirección inicial del tiro modifica el ángulo de salida del tiro de una forma también incuestionable:

$$Posición_{Horizontal\ nueva} = Posición_{Horizontal} + \left[\frac{Dirección}{4} \right]$$

COLISIÓN CON LOS PINOS

Para las colisiones, se usa otra hebra de proceso, representada por otro “timer” que generalmente se encuentra inactivo. Al momento en que se detecta una colisión con un pino de Bowling, se registra el ángulo inicial de colisión y el programa sigue trabajando normalmente:

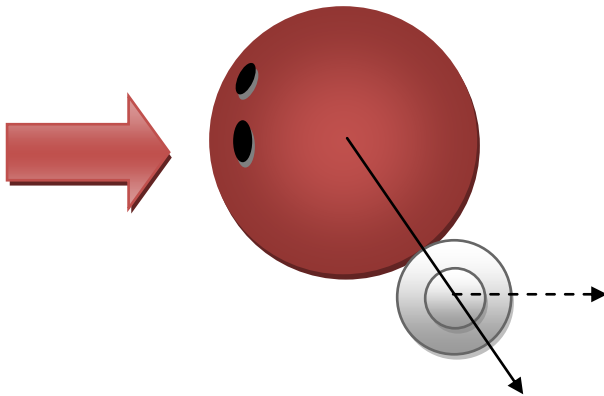


Ilustración 14: Colisión de la bola con un pino

En una colisión, el pino entonces caerá directamente hacia el vector generado por el centro de la bola y del mismo pino. En este formato de colisión simplificado, el mismo “motor” de colisiones que se ejecuta en segundo plano calculará la inclinación hacia el lado de la caída y verificará si esa caída “bota” otro pino que aún no haya colisionado con la bola u otro pino en la misma situación.

Los cálculos de colisiones se mantienen hasta que la bola haya alcanzado su posición final.

La verificación de la colisión inicial es sencilla. En el espacio euclídeo de juego, la comprobación sólo se basa en la distancia entre puntos centros de la bola con cada uno de los pinos (distancia euclídea), tal como recordamos de cursos de cálculo básico no es más que:

$$distancia(bola, pino) = \sqrt{(pino_x - bola_x)^2 + (pino_y - bola_y)^2}$$

Ahora bien, ya tenemos claro cuando tocó la bola a un pino, sin embargo: ¿por dónde lo toco?

Este cálculo también es bastante simple:

$$\text{ángulo}(bola, pino) = \arctan\left(\frac{pino_x - bola_x}{pino_y - bola_y}\right)$$

Ahora, ¿cómo colisionan entre ellos?

Las colisiones siempre han sido un tema importante en computación gráfica y simulaciones físicas. De hecho simular un choque totalmente real sería prácticamente imposible. Desde un punto de vista simplificado, tenemos las posiciones de los pinos. Numerados según son posicionados por mi algoritmo:

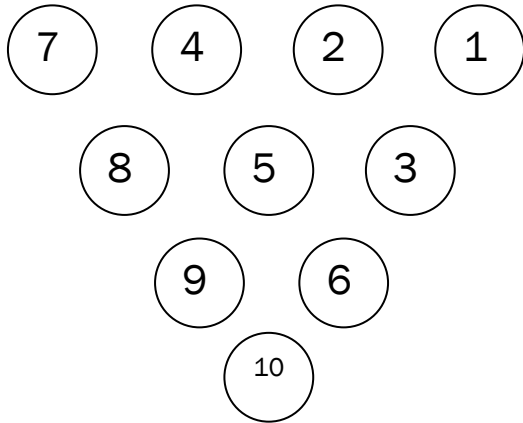


Ilustración 15: Posiciones de los pinos

Así, es curioso que podemos reducir el problema a un ámbito mucho más manejable:

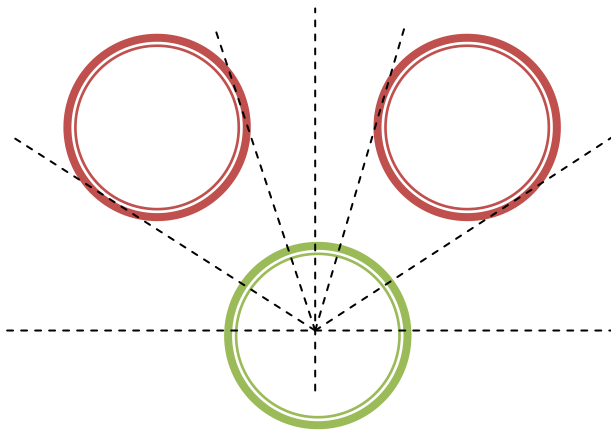


Ilustración 16: Colisión base

Entonces, sólo manejaremos una colisión múltiple y posiblemente encadenada cuando el ángulo en que la bola golpee al pino verde (véase *Ilustración 16*) genere un ángulo que produzca que uno de los pinos rojos caigan (los que a su vez podrían generar el mismo efecto de forma posterior).

Un punto importante es considerar que dado el peso de los pinos, la trayectoria de la bola puede verse afectada, sin embargo en la simulación esto implicaría utilizar diversos tipos de bola por peso y tipos de pinos, además de una multitud de cálculos adicionales sobre los pinos que dificultarían de sobremanera (matemáticamente) las colisiones.

LAS TEXTURAS Y EL ENTORNO

Con formas básicas se puede dar una idea del ambiente, sin embargo si se quiere proporcionar una experiencia envolvente es necesario pulir los detalles y la mejor forma de hacerlo es mediante la incorporación de texturas.

Primero se necesita una base para extraer texturas y que mejor que de un lugar real de juegos como es el Bowling del Mall Curicó Center⁷.

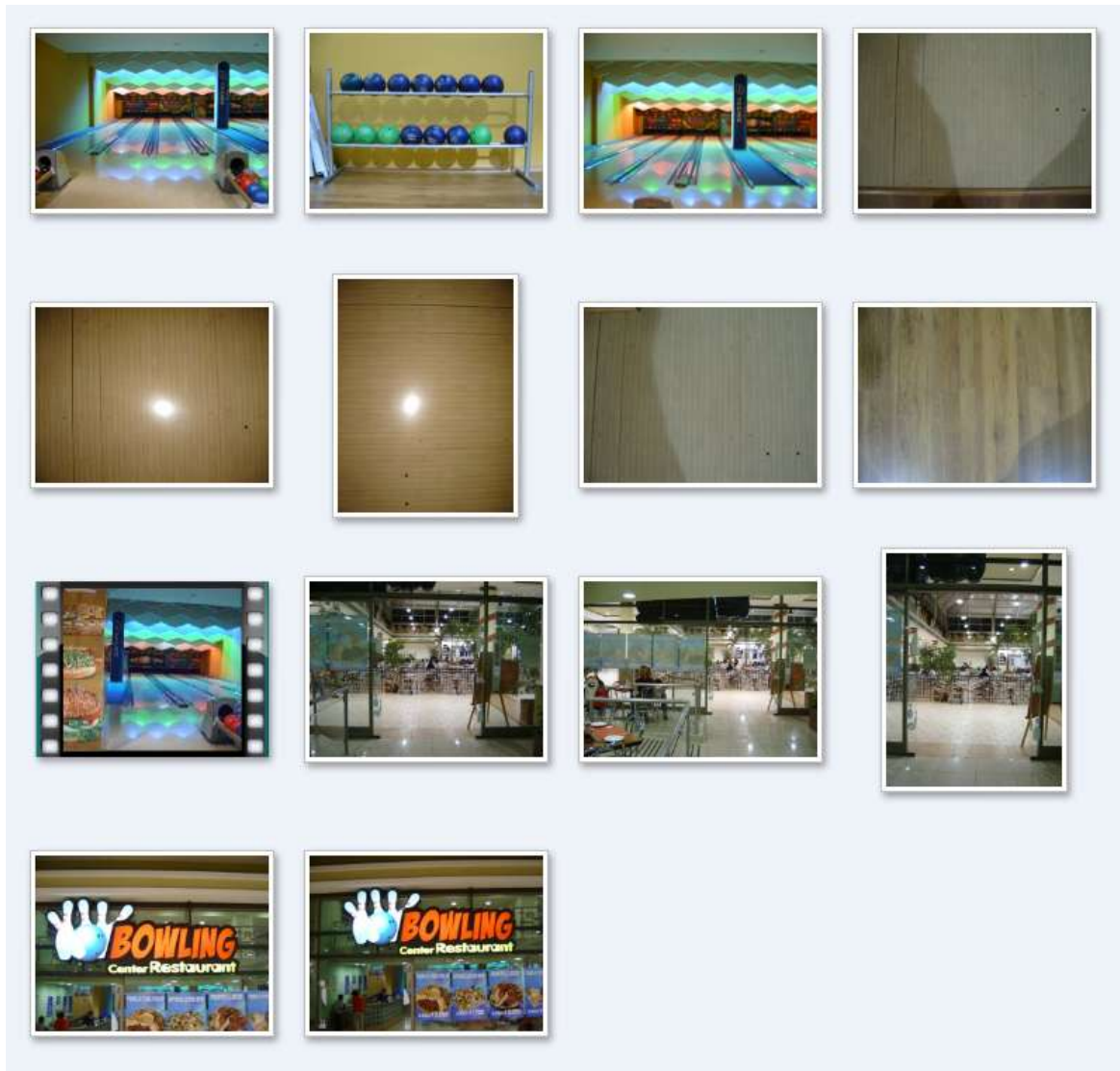


Ilustración 17: Imágenes y texturas utilizadas

Con las imágenes obtenidas se generará el ambiente del Bowling para el trabajo.

⁷ Avda. Bernardo O`Higgins 201 Curicó, Chile

A continuación y a manera de documentación ejemplificaré la forma en que se fueron consiguiendo texturas de alta calidad sin bordes mal alineados ni problemas al repetirlas:

1. En este caso crearemos la textura para la pista. Partimos con la fotografía original capturadas desde una vista superior de la pista:



Ilustración 18: Foto original de la pista

2. Para crear una textura liviana, elegiremos un segmento con el menor número de detalles e imperfecciones:



Ilustración 19: Segmento de pista

3. Preparamos la imagen para la parte final, la misma debe ser ajustada con un ancho y alto idéntico (en este caso 128 píxeles) y con sus características lo más simétricas posibles:



Ilustración 20: Textura en su parte final

4. Con un filtro de "Offset" podemos retocar nuestro segmento:

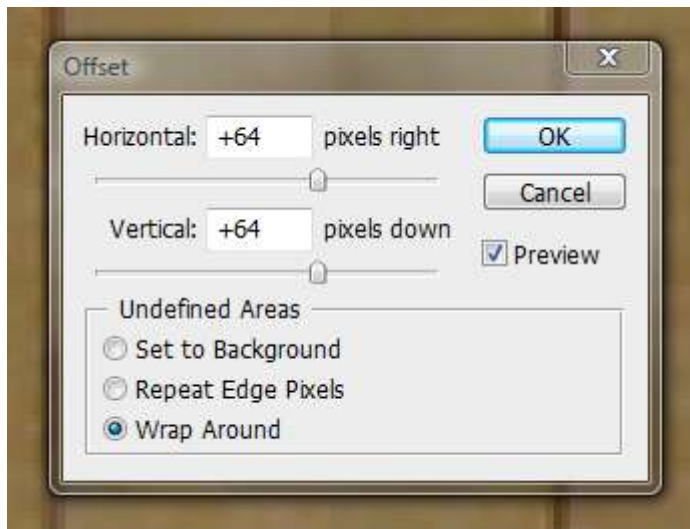


Ilustración 21: Usando el filtro de "Offset"

La idea es imitar la función de repetición en mosaico que hará OpenGL, para así suavizar los bordes de la textura antes de aplicarla al juego.

Así, finalmente podemos obtener una textura de calidad, con la que incluso podemos pintar sin notar los cortes entre tablillas:

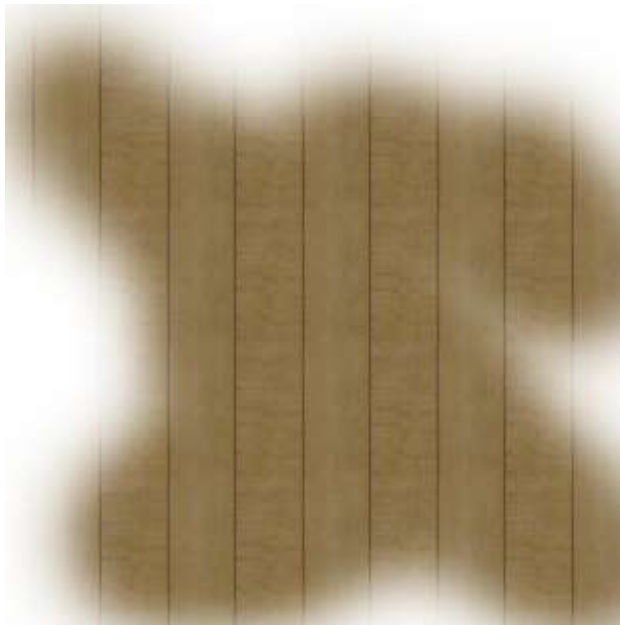


Ilustración 22: Usando la textura final

CONCLUSIONES Y COMENTARIOS FINALES

El programa se desempeña correctamente, aunque faltó bastante funcionalidad para lograr un producto sólido, en general está, a mi parecer bastante completo.

De hecho la parte más importante que fue omitida, fue la funcionalidad como juego en sí, aunque no es más que eliminar algunas configuraciones del usuario y correr el juego con cierto desafío (por ejemplo presionar el botón justo en el poder deseado de una barra que se mueve constantemente).