ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO Y DESARROLLO DE UN JUEGO 3D INTERACTIVO "SQUASH" CON HERRAMIENTAS DE REALIDAD VIRTUAL Y LA LIBRERÍA OPEN SCENE GRAPH"

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del título de:

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACIÓN EN SISTEMAS MULTIMEDIA

Presentado por:

Charles Miguel Pérez Espinoza
Iván Fabricio Salguero Sánchez
Carlos Luis Landívar Borja

Guayaquil – Ecuador

AÑO

2010

AGRADECIMIENTO

Este informe representa el final de una etapa muy enriquecedora de nuestro conocimiento. En toda la experiencia universitaria y en la conclusión del presente trabajo, hay personas que merecen nuestro agradecimiento porque sin su valiosa aportación no hubiera sido posible llegar a donde estamos, y también hay quienes lo merecen por haber plasmado su huella en nuestro camino.

A Dios que es quien nos cuida y protege. A nuestros padres, que fueron nuestra guía y nos brindaron su confianza en la realización de nuestros sueños. Nuestros hermanos que siempre están en el momento justo para ayudarnos. A nuestros amigos con quienes hemos compartido muchos momentos inmemorables que siempre llevaremos en nuestro corazón. A nuestro director PhD. Sixto García por su incansable esfuerzo y empuje que nos brindó para culminar esta ardua tarea.

Gracias a todos, por recordarnos que existen personas valiosas en el mundo.

DEDICATORIA

A nuestros padres, familiares y amigos.

TRIBUNAL DE SUSTENTACIÓN

_____

PhD. Sixto García
PROFESOR DE LA MATERIA
DE GRADUACION

_____

Msc. Federico Raue R.
PROFESOR DELEGADO
DEL DECANO

DECLARACIÓN EXPRESA

La responsabilidad del contenido de este informe de materia de graduación, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la **Escuela Superior Politécnica del Litoral**.

_____

Charles Miguel
Pérez Espinoza

_____

Carlos Luis
Landívar Borja

_____

Iván Fabricio
Salguero Sánchez

RESUMEN

Este informe presenta la implementación de un juego de tenis Squash, por medio del uso de las tecnologías de Realidad Virtual y Multimedia. Teniendo como objetivo principal que los dispositivos de esta índole sirven para crear juegos inmersivos.

La aplicación Squash nació de la idea de generar un juego que genere diversión para grandes y chicos, y que sea lo más real posible, para adaptarlo a futuro a los grandes juegos.

El propósito del juego es tener la práctica de este deporte, golpeando la bola, creando la colisión con la raqueta, del mismo modo que golpee la pared y regrese a la raqueta para golpearla nuevamente; y así sucesivamente para tener un marcador con el número de aciertos y el número total de fallos, dependiendo de las reglas del juego.

Dentro del proyecto utilizamos un dispositivo para rastreo de posición y orientación (tracker), cuyo sensor lo adherimos a una raqueta u objeto similar

para realizar la captura en tiempo real de los movimientos. Y poder traspasarlos a la aplicación como movimientos de raqueta vistos en pantalla. Por ende, si el usuario hace un giro con el objeto que tiene adherido el sensor, la raqueta también gira. Estos movimientos son de forma muy realista, con una mínima de perdida al momento de la lectura.

También fue necesario un proyector con frecuencia de refrescamiento de 120Hz, y un par de gafas estereoscópicas. Ambos implementos necesarios para observar el efecto 3D.

Para comenzar el juego es necesario pegarle a la bola. Su posición inicial es en la mitad de la cancha, y esta va rebotando en el mismo lugar hasta que se detecte una colisión. Cabe recalcar, que a mayor la velocidad del sensor para golpear la bola, esta última obtendrá mayor velocidad.

Otro punto importante es el poder ver la posición de la bola y la raqueta. Fue necesario crear sombras en el piso de la cancha, para no perder el sentido de profundidad de estos objetos y saber exactamente donde se encuentran.

Respetamos todas las reglas del juego Squash real, pero hicimos ciertos ajustes para que el juego sea más amigable como minimizar la gravedad, para que la bola rebote más lentamente.

Obteniendo en sí la aplicación deseada llamada Squash.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Al inicio de la materia de grado nos preguntábamos acerca de lo que haremos, aprenderemos ya que sin duda alguna todo lo que nos ofrecía esta materia era completamente nuevo para nuestro conocimiento. Nos asombramos al ver los equipos que íbamos a usar, o los conceptos nuevos que íbamos a probar e inclusive las librerías y lenguajes que teníamos que aprender para desarrollar algo que realmente demuestre la llamada Realidad Virtual.

Y nos dimos cuenta, poco a poco, de lo que es la Realidad Virtual. Un sistema o interfaz informático que genera entornos sintéticos en tiempo real. Es de naturaleza ilusoria, pues se trata de una realidad perceptiva sin soporte objetivo, sin red extensa, ya que existe sólo dentro del computador.

Después de entender la utilidad de la realidad virtual, y aprender el uso y concepto de los dispositivos que íbamos a usar, fue que pensamos en elegir un tema que sea divertido, inmersivo y sea de fácil uso, que nos permita experimentar cosas nuevas no solo para nosotros sino para grandes, jóvenes y niños por tal motivo es que en esta ocasión y para el informe hemos escogido

su aplicación en un juego de video. De esta manera nos permitirá utilizar y acercarnos más a la tecnología de la realidad virtual tanto en hardware como en software, y a la vez nos permita la utilización de nuestros conocimientos en el uso de diferentes medios.

El juego de video que decidimos escoger para el desarrollo y diseño es el llamado "Squash". El Squash es un deporte de raqueta que se practica en interiores con 2 o 4 jugadores y una pelota de goma que puede tener distintos grados de velocidad o rebote. Los jugadores golpean la pelota con sus raquetas haciéndola rebotar en la pared frontal de la cancha. Pero nuestra aplicación solo se trata de un juego de práctica de un solo jugador.

Decidimos que la implementación sería lo más realista en cuestiones de física para los golpes. Esto ayudará a aumentar el reto. Utilizamos como mando de nuestra aplicación una raqueta a la cual adaptaremos un sensor de movimiento, el cual nos permitirá traducir nuestros movimientos en el mundo real al escenario virtual en tiempo real, teniendo en cuenta la escala correspondiente.

El juego se propone verificar la factibilidad en el uso de la realidad virtual para un juego, que resulte natural y familiar. Se plantea crear un ambiente virtual 3D adecuado, que sea interactivo e inmersivo. Pero lo más importante que permita ejecutar la práctica de este deporte de manera sencilla, con una adecuada sincronización entre las acciones del usuario. Y en este trabajo se explica cómo lo pudimos conseguir.

CAPÍTULO I
CONCEPTOS PRELIMINARES

Antes de todo entendamos un poco que es la realidad virtual, es un sistema o interfaz informático que genera entornos sintéticos en tiempo real, representación de las cosas a través de medios electrónicos o representaciones de la realidad, una realidad ilusoria, pues se trata de una realidad perceptiva sin soporte objetivo, sin red extensa, ya que existe sólo dentro del ordenador. Por eso puede afirmarse que la realidad virtual es una pseudorrealidad alternativa, perceptivamente hablando.

La realidad virtual ha sido desarrollada desde diferentes áreas del conocimiento entre ellas la informática, las matemáticas, la física, la ingeniería espacial, pero, ha sido la primera de ellas la más conocida en cuanto a su generación y progreso.

En los últimos años los avances tecnológicos en muchas áreas han facilitado el desenvolvimiento de las actividades cotidianas, y muchos de estos avances han tenido como pilar fundamental en la Computación, la cual ha alcanzado un desarrollo impresionante en las últimas décadas.

Es así que su apoyo a las diferentes áreas ha permitido la consecución de metas que parecieron inimaginables y consideradas como parte de la ciencia ficción hace muchos años atrás. Su aplicación, aunque centrada inicialmente en el terreno de los videojuegos, se ha extendido a otros muchos campos, como la medicina, simulaciones de vuelo, etc.

Por tal motivo es que en esta ocasión y para este informe hemos escogido su aplicación en un juego de video. De esta manera nos permitirá utilizar y acercarnos más a la tecnología de la realidad virtual tanto en hardware como en software, y a la vez nos permita la utilización de nuestros conocimientos en el uso de diferentes medias.

En este documento hablaremos sobre el desarrollo y diseño de un juego específico llamado “Squash”. El Squash es un deporte de raqueta que se practica en interiores con 2 o 4 jugadores y una pelota de goma que puede

tener distintos grados de velocidad o rebote. Los jugadores golpean la pelota con sus raquetas haciéndola rebotar en la pared frontal de la cancha.

Analizando el tema podemos concluir que la realidad virtual constituye una herramienta totalmente nueva de interacción y de inmersión en nuestro medio siendo este uno de los puntos principales en los que se basa el desarrollo de un juego de video.

Entendamos por juego de video como un programa informático creado para el entretenimiento, basado en la interacción entre una o varias personas y opta por un aparato electrónico que ejecuta dicho videojuego; este dispositivo electrónico puede ser una computadora, un sistema arcade, una videoconsola, un teléfono móvil, teniendo en cuenta que el juego debe ofrecer retos constantes y ser divertidos capturando y manteniendo la atención del jugador en todo momento, permitiéndole a este ser parte del juego convivir con el mismo mostrando realismo e historia que le permita sumergir sus emociones y sentimientos.

Tomando en cuenta lo dicho con anterioridad el problema básico es encontrar una historia, un reto, que capture al usuario y que lo mantenga "pegado al

asiento", además está el hecho que para producir una mejor inmersión debemos emplear esta nueva tecnología que tenemos a disposición como lo es la realidad virtual.

A la mayoría de personas realmente les interesa la tecnología (Realidad Virtual) y como usarlo, como jugarlo y si realmente tiene un parecido a jugarlo en la realidad.

Para encontrar una solución al problema expuesto en los párrafos anteriores hemos recurrido a un deporte, ya que este no contiene historia que sustente nuestra aplicación, pero si un reto muy elevado; que si es desarrollado de la forma indicada podremos mantener la atención del usuario como se desea.

El problema era saber que deporte nos permitiría un mejor y mayor provecho de los dispositivos usados para demostrar la realidad virtual por lo cual nos enfocamos en el tenis.

En este caso optamos por los primeros tipos de juegos en el cual el reto iba en aumento conforme transcurría el tiempo, pero el reto era llegar a un nivel

superior que nadie había logrado o un score que nadie había alcanzado previamente.

De esta manera llegamos a la conclusión de que la aplicación se basaría en el juego de tenis pero no el convencional sino el llamado Squash. Optamos por este juego ya que tampoco existen muchos en el mercado.

Por lo expuesto, decidimos emprender un alcance a nuestro proyecto y crear una aplicación que satisfaga las necesidades de los usuarios basándonos en un juego de práctica del Squash; de un solo jugador, que contenga un score y donde la dificultad será media para que este se pueda practicar.

Además decidimos que la implementación sería lo más realista en cuestiones de física de los golpes esto ayudara a aumentar el reto; utilizamos como mando de nuestra aplicación una raqueta a la cual adaptaremos sensores de movimiento los cuales nos permitirán traducir nuestros movimientos en el mundo real al escenario virtual en tiempo real, teniendo en cuenta la escala correspondiente.

Nuestra intención con este tipo de proyecto es llegar a usuarios de diversas edades tanto hardcore como casuales, estrategia que al magnate de los videojuegos Nintendo™ le ha funcionado hasta hoy. Se lo puede jugar con gafas estereoscópicas, activas (las de cristal líquido usadas) o pasivas (anaglíficas), junto con un proyector de 120 Hz de frecuencia de actualización para las primeras, o con uno convencional para las últimas.

Siempre y cuando teniendo como objetivo general el desarrollar un juego de práctica "squash" que permita una buena inmersión, que capture la atención del usuario usando dispositivos para demostrar la realidad virtual.

Y obteniendo objetivos específicos como el desarrollo de un juego con un reto de nivel medio apto para todo tipo de jugadores. Hacer que los movimientos del balón dentro del juego nos brinde el realismo apropiado.

Dentro de las metas del seminario del uso de la tecnología de realidad virtual en sistemas multimedia era utilizar herramientas de este tipo para controlar sistemas que hemos aprendido a lo largo de nuestra carrera. Y al observar y haber aprendido tantos sistemas importantes, quisimos darle un toque de novedoso a lo que naturalmente creamos y fue desarrollar un juego tal que no solo use dispositivos multimedia sino algoritmos de juegos de video y

poder mostrar que si es posible lograr realismo con estos tipos de programas, que deben ser 100 % inmersivos para que puedan ser aceptados por los usuarios que vayan a utilizarlo.

Además se utiliza 2 dispositivos importantes para la inmersión que fueron considerados muy importantes para el desarrollo de juegos, como son los sensores de movimiento, el proyector y gafas estereoscópicas.

Elegimos el juego Squash para poder demostrar que con los dispositivos usados para la realdad virtual se puede tener movimientos reales como el movimiento de la raqueta o el rebote de la bola.

Optamos por este tipo de juego en el cual el reto iba en aumento conforme transcurría el tiempo, pero el reto era llegar a un nivel superior que nadie había logrado o un score que nadie había alcanzado previamente, de esta manera llegamos a la conclusión de que la aplicación se basaría en el juego de tenis pero no el convencional sino en un squash donde el culpable de la perdida éramos nosotros mismos.

Al haber seleccionado un deporte como es el squash nos aseguramos que gente joven como adulta se permita vivir esta experiencia. A futuro en otras versiones convertir a este juego en una opción familiar que cualquier padre aprobaría para sus hijos y cualquier adulto jugaría con sus hijos.

El presente trabajo se propone verificar la factibilidad en el uso de la realidad virtual para un juego, que resulte natural y familiar. Se plantea crear un ambiente virtual 3D adecuado, que sea interactivo e inmersivo, que permita ejecutar una práctica de este deporte sencillo, con una adecuada sincronización entre las acciones del usuario, la retroalimentación visual y el sonido generado. Además se analizaran los principales factores que permitan que la interfaz se convierta en un ambiente agradable para jugarlo.

CAPÍTULO II
ANÁLISIS DE LA SOLUCIÓN

2.1. Herramientas usadas para el análisis

2.1.1. Encuesta

Como anteriormente hemos explicado, hicimos una encuesta a 50 personas para poder tener un campo de solución o un alcance de nuestro proyecto.

La encuesta se trataba de 9 preguntas que resumían si era bueno un proyecto de Squash y la tecnología a usarse. Y a continuación se demuestra la tabla de respuestas:

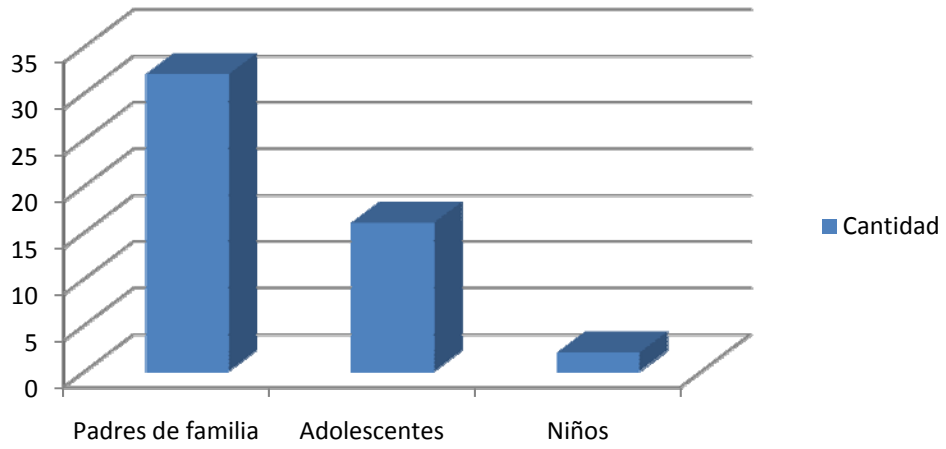


Fig. 1: Número y tipos de encuestados

Separamos a los encuestados, ya que solo nos interesaban padres de familia, adolescentes y niños en nuestra investigación. Los separamos por dos preguntas, la edad y si tiene hijos o no tiene hijos (si eran mayores de edad), o si eran jóvenes el rango de edad y separábamos niños o adolescentes:

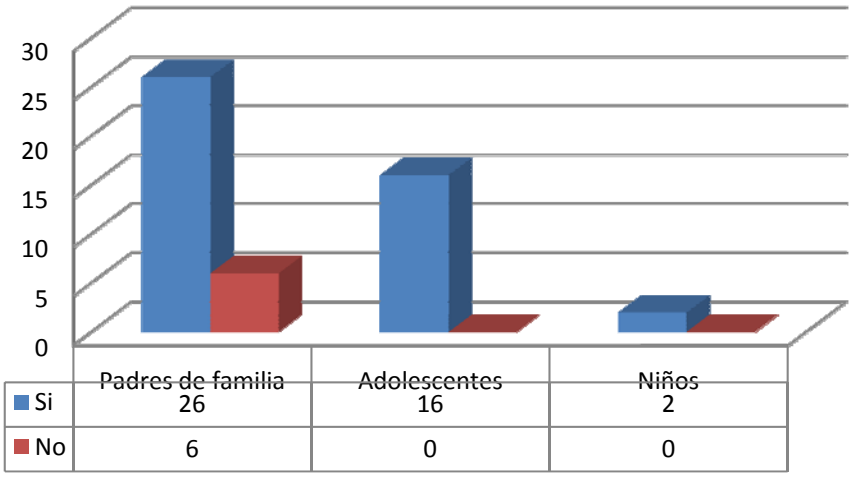


Fig. 2: ¿Usted conoce la tecnología de realidad virtual?

Esta pregunta nos hizo entender si la tecnología es conocida entre las personas, si era necesario explicarla de forma más detallada, por los resultados obtenidos los niños y adolescentes si sabían al 100% lo que es esta tecnología, pero en los padres de familia existía un pequeño grupo que tuvimos que explicar un poco más detallado el tema.

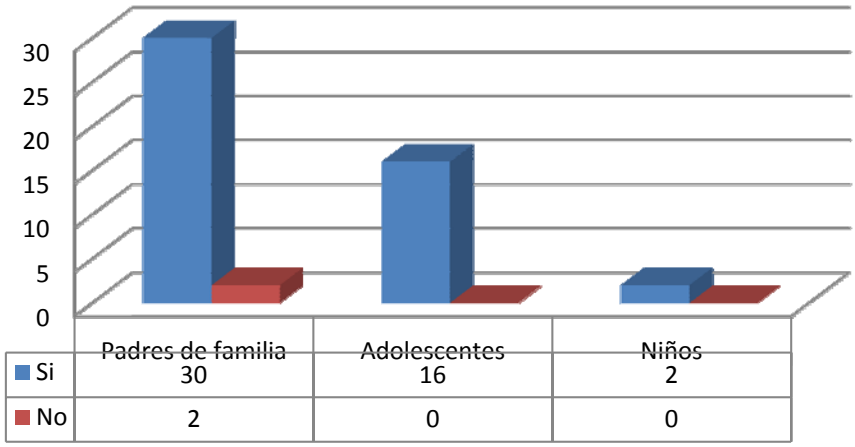


Fig. 3: Después de observar las imágenes anaglíficas, y tuviera la oportunidad de utilizar esta tecnología, ¿la usaría?

En esta pregunta analizamos si las personas están interesadas en esta nueva tecnología, si realmente los guayaquileños estaban dispuestos a usar esta tecnología, si lo que estamos haciendo como proyecto valía la pena. Y hubo mucha aceptación, aunque dos personas si se les hizo difícil entender para que realmente usan esta tecnología.

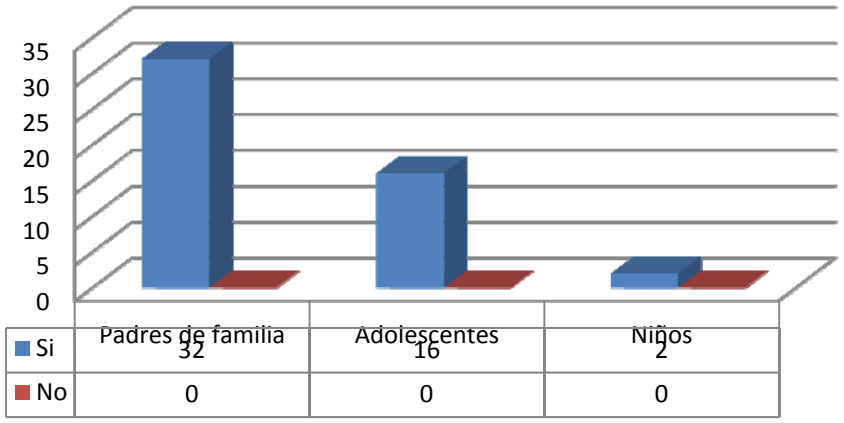


Fig. 4: En el área del entretenimiento, ¿usted estaría dispuesto a experimentar una propuesta con esta tecnología?

Al agregar el tema de entretenimiento el 100% de los encuestados, les gustó la idea de usar la tecnología de realidad virtual, ya que hicimos mención de las películas en 3D que ahora se ven en los cines, y si hubiera la oportunidad de que la mayoría de serie de televisión y juegos sean en 3D.

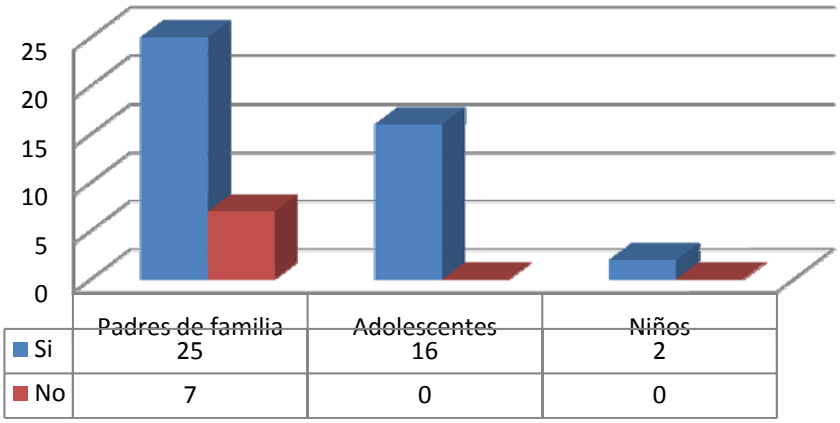


Fig. 5: ¿Estaría dispuesto a invertir en un juego que pudiese compartir con sus hijos?

En el caso de los adolescentes y los niños cambiamos a la pregunta de que si estarían dispuestos a decirles a sus padres que les adquieran un juego con la tecnología de realidad virtual. Los padres de familia estarían dispuestos a invertir porque la mayoría tenían hijos menores de edad, y los otros tenían a sus hijos ya mayores de edad.

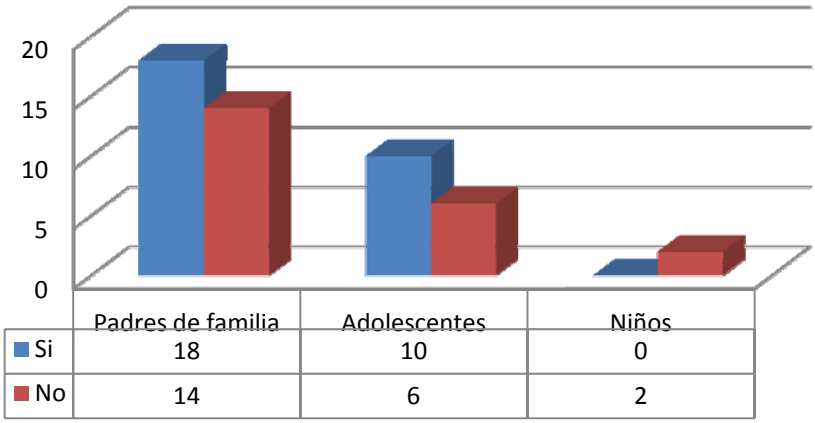
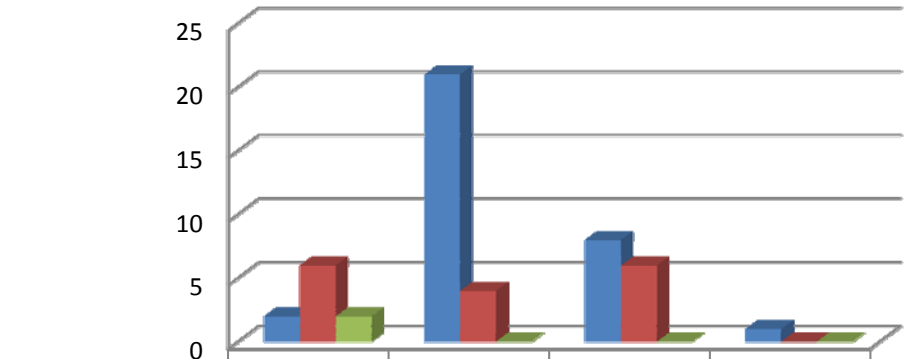


Fig. 6: ¿Estaría dispuesto a jugar Squash en la sala de su casa utilizando realidad virtual? (en esta pregunta se hizo una breve explicación acerca del juego)

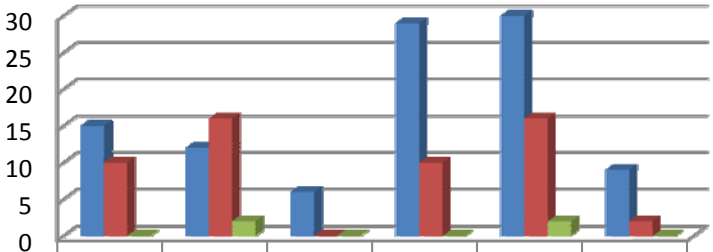
A la mayoría de encuestados se tuvo que explicar lo que era el juego de Squash, ya que en Ecuador este juego no es tan conocido, de igual forma en los padres de familia que tenían hijos mayores de edad no estaban dispuestos a jugarlo, y los otros que dijeron que no fue porque tenían hijos recién nacidos o máximo 3 años de edad, en los adolescentes fue distinto ya que algunos preferían juegos de acción más que de deporte, y a los niños no les interesaba para nada.



	($) 1 - 2.99	($) 3 - 4.99	($) 5 - 10	($) 10 +
Padres de familia	2	21	8	1
Adolescentes	6	4	6	0
Niños	2	0	0	0

Fig. 7: Y si es así; ¿Cuánto usted estaría dispuesto a pagar por una entrada a un sitio que le permita experimentar un juego así?

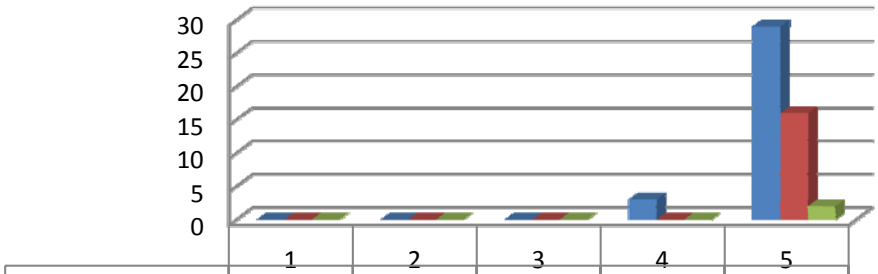
En esta pregunta analizamos los precios que las personas estarían dispuestos a pagar, la mayoría de padres de familia pagaban dentro del rango de 3 a 5 dólares, y los adolescentes lo mismo, ya que les parecería nuevo e interesante poder tener salas de juegos de este tipo.



	Medicina	Juegos	Spa	Turismo	Peliculas	Otros
Padres de familia	15	12	6	29	30	9
Adolescentes	10	16	0	10	16	2
Niños	0	2	0	0	2	0

Fig. 8: ¿En qué áreas usted cree que también le gustaría que se use la realidad virtual?

Esta pregunta era de opción múltiple se podía elegir algunas opciones, lo cual algunos pusieron otras opciones como las películas para adultos (eróticas o similares a esta), pero al final se decidió mucho por el turismo, ya que aumentaría las visitas de muchos turistas, o en películas que sería más impactante, y en medicina ya que en otros países ya se la usa sería bueno que aquí también se la use.



	1	2	3	4	5
Padres de familia	0	0	0	3	29
Adolescentes	0	0	0	0	16
Niños	0	0	0	0	2

Fig. 9: ¿Cree usted que la realidad virtual mejoraría el realismo y la inmersión de los juegos? (ponga del 1 al 5, 5 siendo la puntación máxima)

Dentro de estos resultados analizamos que la mayoría de estas personas les interesa mucho lo que es la tecnología de Realidad Virtual por ende estábamos 100% seguros de crear este proyecto de graduación.

2.1.2. Los implementos

Los implementos que se deben usar para este proyecto son: la bola y la raqueta, lo cual nos permitió crear las colisiones y poder jugar de forma normal. Ya que el juego de Squash se lo juega con estos implementos.

Fig. 10: Raqueta de squash

2.1.3. El escenario

Otra parte importante del análisis, fue la creación del escenario que debería ser con las medidas exactas (escaladas), para poder crear las reglas de igual forma que el juego Squash real. Se debía crear el escenario de tal forma que debe parecer real para mantener inmersivo el juego.

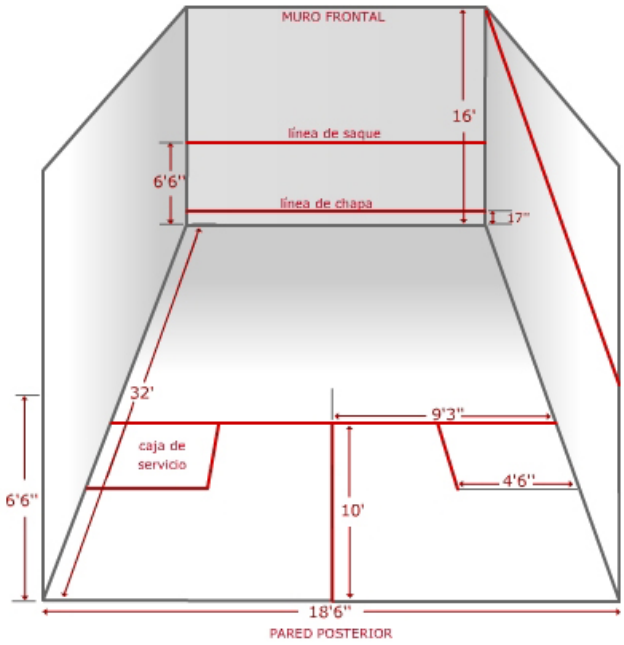


Fig. 11: Medidas oficiales de una cancha de squash (en pies y pulgadas)

2.1.4. Las reglas

Dentro del análisis de la solución que tomamos era basarnos en las reglas del Squash convencional por ende leyendo en el manual de este juego, aplicamos las siguientes reglas a nuestro proyecto:

Reglas del saque

El saque comienza una vez que la raqueta impacta la bola.

El golpe debe ser directo a las paredes sin tocar el piso.

Al pegar en la pared frontal, el saque es válido siempre que la bola golpee entre la línea de servicio y la línea del tin.

El jugador tiene opción a dos saques antes de que el juego considere que falló.

Reglas del Juego

Una vez hecho un saque válido el juego comienza permitiendo que la bola rebote una vez en el piso antes de que se la golpee con la raqueta, caso contrario es una falla.

Se es permitido que la bola golpee contra las paredes cualquier cantidad de veces antes de golpear con la pared frontal.

Se considera una falla cuando golpea en las paredes si la bola golpea sobre las líneas de out.

Se considera falla también cuando golpee en la pared frontal por debajo del tin.

Se marca un acierto siempre que golpee con la pared frontal sin que ocurriese una falla.

2.1.5. La Física

Otros de los análisis importantes que hicimos fue la física que íbamos a usar para crear la realidad del juego, como son las colisiones, la gravedad, el impulso a pegar, la normal que se forma al momento de hacer la colisión la bola con la pared o con la raqueta, y así algunas fórmulas que se detallaran a continuación:

Rebote con las paredes y piso

Dado que las paredes y piso están todas en planos ortogonales respecto al origen, el rebote de la bola se simplifica significativamente. De los 3 componentes de la velocidad de la bola, sólo el que es perpendicular de colisión, cambia de sentido, los otros dos siguen igual.

Pero sí se toma en cuenta el coeficiente de restitución del impacto entre dos objetos inelásticos. Considerando que la masa de la bola es

despreciable con respecto al piso o pared, la nueva magnitud de cada componente de la velocidad queda así:

$$|V_{nueva}| = |V_{anterior}| * C_R \qquad (Ec.1)$$

Donde CR es el coeficiente de restitución. Así, con cada rebote, la bola va perdiendo un poco de su velocidad.

Rebote con la raqueta

Para detectar la colisión entre la bola y la raqueta, se utiliza la esfera de intersección que se encuentra alrededor de la raqueta. Si la bola está dentro de ésta, entonces se calcula la distancia entre el centro de la bola y el plano de colisión, paralelo a la malla de la raqueta. Cuando esta distancia pasa de ser positiva a negativa, indicando que la bola traspasó el plano, la bola es reposicionada pegada al plano antes de ser dibujada en ese cuadro, y ahí es calculada la nueva dirección y magnitud de la velocidad de la bola. Adaptando el procedimiento dado en [4] para una colisión tridimensional, entonces la velocidad de la bola tiene 3 componentes:

$$V_{bola} = V_{\perp} + V_{tang1} + V_{tang2} \qquad (Ec.2)$$

Donde $V_{\perp}$ es el componente perpendicular al plano de colisión, y V_{tang1} y V_{tang2} son tangentes al plano, y no paralelos. En el caso de la raqueta, la velocidad del punto de impacto en sí es dada por:

$$V_{pi} = V_{piv} + V_{rm} + V_{rp} \qquad (Ec.3)$$

Donde V_{piv} es la velocidad del pivote de la raqueta, V_{rm} es la velocidad dada por la rotación en torno al mango, y V_{rp} la velocidad dada por la rotación en torno al pivote. Cada uno se calcula de acuerdo a la velocidad lineal y angular de la raqueta en el cuadro dado. Y esta velocidad es luego descompuesta de la misma manera que la bola en la Ec. 2 (normal y 2 tangentes no paralelas del plano), para así poder obtener un nuevo vector de velocidad de la bola muy realista. Cabe recalcar que el componente normal de la nueva velocidad de la bola es calculado con coeficiente de restitución, tomando en cuenta la masa de la raqueta y de la bola. Pero sólo la bola es afectada por el cambio de velocidad y no la raqueta, por motivos de simplicidad y rendimiento. Entonces:

$$V_{nb2} = \frac{(C_R+1)M_r V_{nr1}+V_{nb1}(M_b-C_R M_r)}{M_b+M_r} \qquad (Ec.\ 4)$$

Donde V_{nb2} es la velocidad normal final de la bola, V_{nb1} es la velocidad normal inicial de la bola, V_{nr1} es la velocidad normal inicial de la raqueta, M_b es la masa de la bola, M_r es la masa de la raqueta, y C_R es el coeficiente de restitución (0 a 1).

La gravedad

La gravedad es la aceleración que experimenta un objeto hacia el centro de la tierra, es lo que nos mantiene en el piso. Esta aceleración es fundamental para el juego de Squash, ya que al momento de rebotar contra la pared, hace que la bola caiga al piso y se le pueda pegar una vez más, ya que si esta gravedad es 0 la bola en si seguiría en el aire con solo la normal de la colisión de la pared y no con la que se dirige hacia abajo.

Por ende la gravedad es un vector más que vamos a usar en nuestro proyecto para el movimiento de la bola.

2.1.6. Profundidad

La profundidad del juego debía ser también algo primordial dentro del juego, hacer que la posición de la cámara, sea de tal manera que se pueda observar las tres dimensiones, la posición de la bola y la raqueta, para poder saber el momento y el lugar donde golpear.

Una de las cosas importantes que se pensó hacer es la sombra de cada objeto y el sonido cuando se golpea, para que se tenga una percepción de saber dónde está la bola, o sea saber la profundidad donde se encuentre, si es por delante de la raqueta o por detrás de la misma, a la izquierda o la derecha.

Teníamos que fabricar una fuente de luz por arriba de estos objetos para que la sombra se pueda observar en el piso y saber la posición de las cosas.

Este fue otro análisis importante para el proyecto.

CAPÍTULO III
ANÁLISIS DE REQUERIMIENTO

En el desarrollo de nuestro software hemos optado por un desarrollo en espiral. Con este alcanzamos pequeñas metas, analizamos, desarrollamos y probamos, para nuevamente planificar y determinar nuevos objetivos.

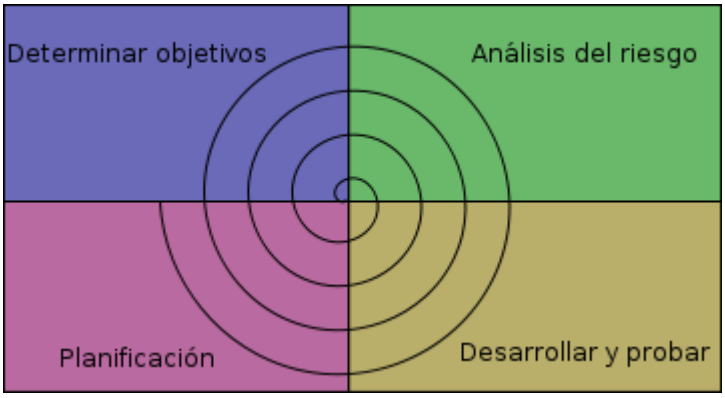


Fig. 12: Proceso del desarrollo en espiral

3.1. Definición del diseño

Al utilizar herramientas nuevas o desarrolladas por otras compañías hemos optado para un óptimo desarrollo del proyecto utilizar librerías que nos permitan un manejo más sencillo y eficiente de los periféricos en este caso nos referimos al manejo de los sensores y el las gafas pasivas para lo cual utilizamos una librería que nos permite una interfaz más sencilla en el manejo de las coordenadas de movimiento.

Teniendo como ventajas que el modelado en espiral puede adaptarse y aplicarse a lo largo de la vida del desarrollo del software, hasta cuando se entrega el software.

Como el software evoluciona, a medida que progresa el proceso, nosotros comprendemos y reaccionamos mejor ante riesgos en cada uno de los niveles evolutivos de errores que se pueden dar.

Nos permitió al momento de desarrollar, que apliquemos el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Y nos redujo los riesgos antes de que se conviertan en problemáticos.

3.2. Requerimientos Funcionales

Se requiere el desarrollo de una aplicación de software que simule por medio de dispositivos de realidad virtual el funcionamiento de una práctica del juego Squash de un solo jugador.

Debe crearse un escenario virtual en tres dimensiones, en el cual deben constar los siguientes objetos de entrada importantes y funcionales:

Raqueta de Squash

Usada para golpear la bola

Bola

Usada para jugar el squash

Tiempo

Usado para saber el cuanto tiempo le queda al jugador para seguir en la práctica y ver si su marcador debe estar en los mejores puntajes.

Y obtener la salida:

Marcador

Usado para demostrar en el momento del juego cuantas veces, se ha fallado y cuantas veces se ha acertado en el golpe de la bola, además es usado para ver si se llega a los mejores puntajes.

3.2.1. Funciones

Para esto se requieren diferentes funciones que nos ayudarán, por medio de los objetos de entrada, llegar a las salidas del sistema. Estas funciones son:

1. La aplicación debe verificar si están conectados los sensores, de otra manera el teclado será necesario para poder jugar el Squash.

2. La aplicación debe permitirle al jugador escoger las opciones que están dentro del menú principal.

3. Al momento de jugar la aplicación debe saber si el golpe de la bola contra la pared es correcto o no lo es, y aumentar el marcador de aciertos o fallas correspondientemente.

4. La aplicación permitirá al jugador mover la raqueta del juego según como tenga la posición y orientación del sensor.

5. La aplicación permitirá grabar el nombre del jugador si es que este ha superado a uno de los marcadores anteriores y aparecer en los diez mejores marcadores.

6. La aplicación debe parar el juego cuando el cronómetro de la práctica haya acabado, sea este de un minuto o de tres, depende lo que el jugador haya escogido.

7. La aplicación permitirá al jugador observar los diez mejores marcadores de aciertos obtenidos hasta el momento.

3.2.2. Casos de Uso y Escenarios

Caso de Uso 1: Comenzar la práctica de Squash

Escenario: Reproducción exitosa del escenario y comenzar a jugar

Actores: Usuario (Jugador)

Supuestos:

- El usuario mueve el ratón en el primer cuadro del menú principal.
- El usuario presiona clic con el botón izquierdo del ratón y espera.

Salidas:

- Se carga la cancha en 3D de Squash.
- El tiempo comienza en 0:00 minutos.
- El marcador de fallos y aciertos se encera.
- Comienza la bola a rebotar en el centro de la cancha.

Descripción:

El usuario comienza el juego dentro de su PC, se coloca las gafas y ve hacia el proyector donde entra al menú principal. Elije el primer o el segundo cuadro del menú (se diferencia entre que se puede practicar 1 minuto o 3 minutos correspondientemente). Ahora espera que la pantalla inicialice tanto el tiempo como los marcadores de fallos y aciertos. Aparece una cuenta regresiva de 3 segundos y la bola

comienza a rebotar en el centro de la cancha, esperando a que el usuario comience a jugar.

Caso de Uso 2: Colisión entre la raqueta y la bola

Escenario: Colisión exitosa entre la raqueta y la bola

Actores: Usuario (Jugador)

Supuestos:

- El usuario mueve el sensor que esta adherido a la raqueta u objeto similar.
- Golpea a la bola.

Salidas:

- La bola se mueve en dirección la cual se obtuvo en la colisión.
- La raqueta queda en el lugar en donde se golpeó la bola.
- La aceleración de la bola aumenta y disminuye a la vez por la gravedad.

Descripción:

Para que el usuario comience a jugar es necesario que golpee a la bola con la raqueta, lo que hace que se cree una cadena de funciones y ecuaciones para saber la nueva velocidad y aceleración de la bola. Al momento de mover la raqueta con el sensor y tocar la bola, se detecta una colisión, y tanto la magnitud como la dirección de la velocidad de la bola cambian según el golpe de la raqueta. Eso no cambia que la bola se vea siempre afectada por la gravedad, que es una aceleración hacia abajo, afectando el vector de velocidad que finalmente afecta a la bola.

Caso de Uso 3: Colisión entre la bola y la pared

Escenario: Colisión exitosa entre la pared y la bola

Actores: Usuario (Jugador)

Supuestos:

- El usuario espera a que la bola rebote contra la pared.

Salidas:

- La bola se mueve en dirección la cual se obtuvo en la colisión de la raqueta y la bola.
- La raqueta sigue el camino según es captado por el sensor.

- La bola golpea con la pared.
- La aceleración de la bola disminuye y un poco más a la vez por la gravedad.

Descripción:

El usuario después de golpear la bola, espera que la bola rebote contra la pared. Ahí se detecta una colisión entre la pared y la bola. De igual forma que cuando pega contra la raqueta, se crean funciones y ecuaciones que resultan en una nueva velocidad para la bola. Así mismo, la gravedad, siempre presente, altera la velocidad final que la bola obtiene.

Caso de Uso 4: Ver puntajes altos

Escenario: Éxito cargo el archivo y lo presenta en pantalla

Actores: Usuario (Jugador)

Supuestos:

- El usuario mueve el ratón en el tercer cuadro del menú principal.
- El usuario presiona clic con el botón izquierdo del ratón y espera.

- El archivo se carga.

Salidas:

- El archivo se ordena por medio de una función.
- Solo aparecen los nombres de las personas con mayor número de aciertos.
- Aparece en pantalla.

Descripción:

El usuario entra al juego y aparece el menú principal. Elije el tercer botón del menú, que es para poder observar los puntajes altos, que están grabados en un archivo texto. Al momento de leer el archivo, la aplicación obtiene los mejores puntajes, aquellos que tengan mayor número de aciertos, y los presenta en la pantalla.

3.3. Requerimientos no funcionales

Estos son los requisitos no funcionales del sistema pero que son importantes para el desarrollo del mismo los cuales incluyen:

Facilidad de uso

El sistema debe ser de fácil uso y entrenamiento por parte de los usuarios del juego, así como de fácil adaptación de la entidad con el mismo. Además el sistema no debe permitir el cierre de una operación hasta que todos sus procesos, subprocesos y tareas relacionados, hayan sido terminados y cerrados satisfactoriamente.

Rendimiento y escalabilidad

El sistema está en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial, ya que dejamos ciertos parámetros dentro del código para que se pueda seguir incrementando ciertas funciones en el mismo.

Normas técnicas y reglas del juego

Las reglas y técnicas del juego son otra parte importante dentro del mismo. Son las que se deben cumplir para saber si al momento de jugar fue una falla o un acierto, según en el lugar donde haya pegado la bola. Indican también el número de rebotes que dará la bola al estar jugando, antes de que se declare una falla.

Percepción de la distancia en la cancha

Es otro requerimiento no funcional, ya que si no percibimos la profundidad del lugar donde se encuentra la bola o la raqueta, se hará muy difícil poder pegarle con precisión a la misma. Por ende la creación de sombras y sonidos fue lo que solucionó e hizo que esta parte sea más sencilla.

3.4. Arquitectura de la Aplicación

Detallamos los procesos que se realizan en cada nivel, y se muestra la arquitectura del sistema.

3.4.1. Nivel de interface gráfico del usuario

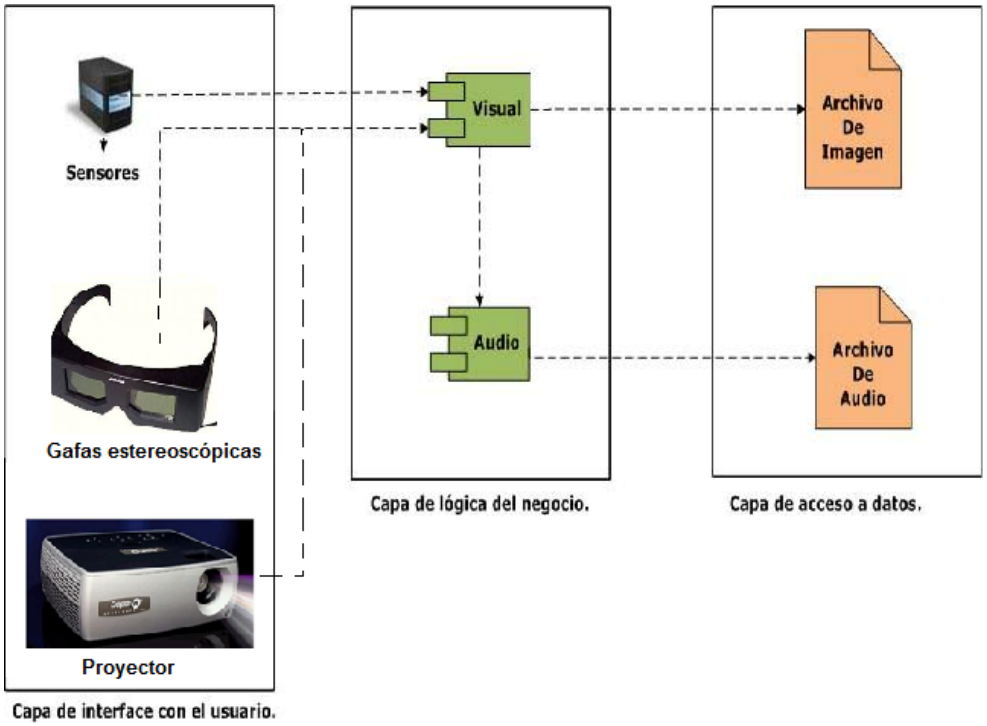


Fig. 13: Nivel de Interface Gráfica con el usuario

Carga de escena 3D

Por medio de la librería Open Scene Graph (OSG) cargamos la escena tridimensional: la cancha de Squash, raqueta y bola.

De igual forma se ejecutan las letras y los archivos para ver el puntaje.

Y usando las gafas estereoscópicas y el proyector podremos ver la cancha.

3.4.2. Nivel de Lógica de la aplicación

Obtención de Coordenadas

Con la librería propia del dispositivo de tracking Polhemus Liberty (PDI), obtenemos las coordenadas X, Y y Z que registran los cambios de movimiento en la raqueta; estas coordenadas son usadas para mover el objeto que se representa en la escena virtual, y se tiene que llegar a tener muchos aciertos para poder tener su nombre en el menú de puntajes altos.

Retroalimentación Visual

Después del movimiento del sensor (adherido al wii-mote), se ve el movimiento escalado de la raqueta, para que haga una colisión con la bola, así mismo se ve que la bola golpea la pared y rebota para el lugar donde está la raqueta.

3.4.3. Nivel de acceso a datos

Lectura de los archivos de puntajes.

Por medio de esta capa obtenemos el archivo (.txt) asociado con los puntajes.

Lectura de los archivos de sonido.

Por medio de esta capa obtenemos el archivo (.wav y .ogg) asociado con la canción inicial y las colisiones.

3.5. Herramientas para crear la solución

Para el desarrollo de nuestra propuesta de un juego utilizaremos las siguientes herramientas:

OpenSceneGraph (OSG)

Es una librería de código abierto, de alto rendimiento para gráficos 3D. Basada en la librería OpenGL y utiliza el ANSI C++, es por esto que trabaja en OSX, GNU/Linux, IRIX, Solaris. HP-Ux, AIX y FreeBSD. Es muy utilizada para la creación de ambientes virtuales, ya que soporta

una amplia gama de imágenes 2D y modelos 3D, en formatos tales como OBJ, 3DS, JPEG, PNG, TIFF, entre otros. Esta librería está en constante actualización por lo cual las versiones varían en muy poco tiempo. Actualmente está en la versión 2.9.6. En el presente proyecto usamos la 2.8.2, creada en julio de 2009, pues esta versión acepta los drivers de los dispositivos usados para el juego.

Se crean diferentes esquemas y algoritmos en el lenguaje C++ que ayudan a crear nodos que contienen las diferentes imágenes o polígonos para que con la combinación de estos crear aspectos casi reales de cada artículo u objeto que se encuentre en el ambiente. Además de utilizar ecuaciones físicas de caída libre con la gravedad y fórmulas matemáticas para crear el movimiento parecido o igual al de la realidad.

Fig. 14: Objetos creados con OpenSceneGraph

Usamos OSG porque podíamos utilizar diferentes librerías, como la SDL que se acoplaba fácilmente a este código. Igualmente las librerías de los trackers, necesarias para la captura de movimiento.

Existen otras librerías, como la OpenGL, pero era tan básica que no incluía muchas de las funciones que OSG ya nos brindaba.

3D Studio Max™

Herramienta propicia para el modelado de objetos en tres dimensiones, cuya extensión (.3ds) es soportada por OSG.

Usamos el 3D Studio Max porque al momento de crear el objeto y guardarlo se lo podía hacer con la extensión .3ds, que era necesaria

para el uso en OpenSceneGraph. Y teníamos más experiencia en usar este programa de modelamiento tridimensional que de los otros.

Visual Studio .Net 2008 (C++) ™

IDE desarrollado por Microsoft para el desarrollo de aplicaciones en diferentes lenguajes, incluyendo, C++.

Usamos el Visual Studio .Net 2008, porque fue una recomendación dentro de la página de OpenSceneGraph, ya que era mucho más sencillo el uso de esta librería.

Dispositivos de entrada y salida

Los dispositivos usados en el proyecto son los que realmente le dan la inmersión al juego de Squash, ya que no sólo la programación del ambiente virtual basta, sino la realidad con la que se debe jugar como son los movimientos, gravedad, velocidad, cancha, etc.

Como opción en el seminario tenemos un proyector *DepthQ*™ con una frecuencia de actualización de 120 HZ, que junto a la tarjeta gráfica

Nvidia™ QUADRO™, permite emitir una imagen que a simple vista se ve duplicada. Para poder ver el efecto 3D, el jugador necesita usar unas gafas estereoscópicas activas, las cuales reciben una señal de sincronización por parte de un emisor infrarrojo conectado directamente a la tarjeta gráfica.

Fig. 15: Proyector DepthQ™ estereoscópico

Fig. 16: Tarjeta Nvidia™ QUADRO™ de 512MB

Para captar el movimiento del jugador fue necesario el uso del sistema de captura de movimiento Polhemus Liberty™. El equipo usado consta de un rastreador magnético, junto al emisor de señal y la fuente de poder. Éstos captan y envían al computador en tiempo real su posición, y así poder graficar el movimiento en el juego. Se incluye también un disco con instaladores de aplicaciones de prueba y diagnóstico, y de las librerías propietarias usadas en el desarrollo de la aplicación.

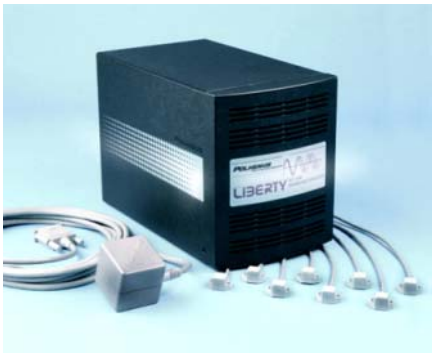


Fig. 17: Rastreadores con su emisor (a la izquierda) y la fuente de poder (derecha), del equipo Polhemus Liberty™

Los valores dados por los rastreadores son enviados por medio de una conexión USB entre la fuente de poder (donde también se procesan las señales) y el computador. Estos datos pueden ser usados en el proyecto gracias a la librería propietaria dada por el fabricante. La precisión de los datos de posición de los rastreadores depende de la distancia entre éstos y el emisor, ya que tienen un alcance omnidireccional máximo de 1,5 metros. Esto fue un problema al principio ya que limitaba los movimientos libres del juego, pero fueron corregidos porque se escaló los movimientos.

CAPÍTULO IV
DISEÑO

4.1. Especificación de casos de uso

Finalizado el análisis de la solución, ya es posible especificar aún más los distintos casos de uso. A continuación los diagramas de interacción respectivos, junto a una descripción de los objetos involucrados en cada caso.

Caso de Uso 1: Comenzar la práctica de Squash

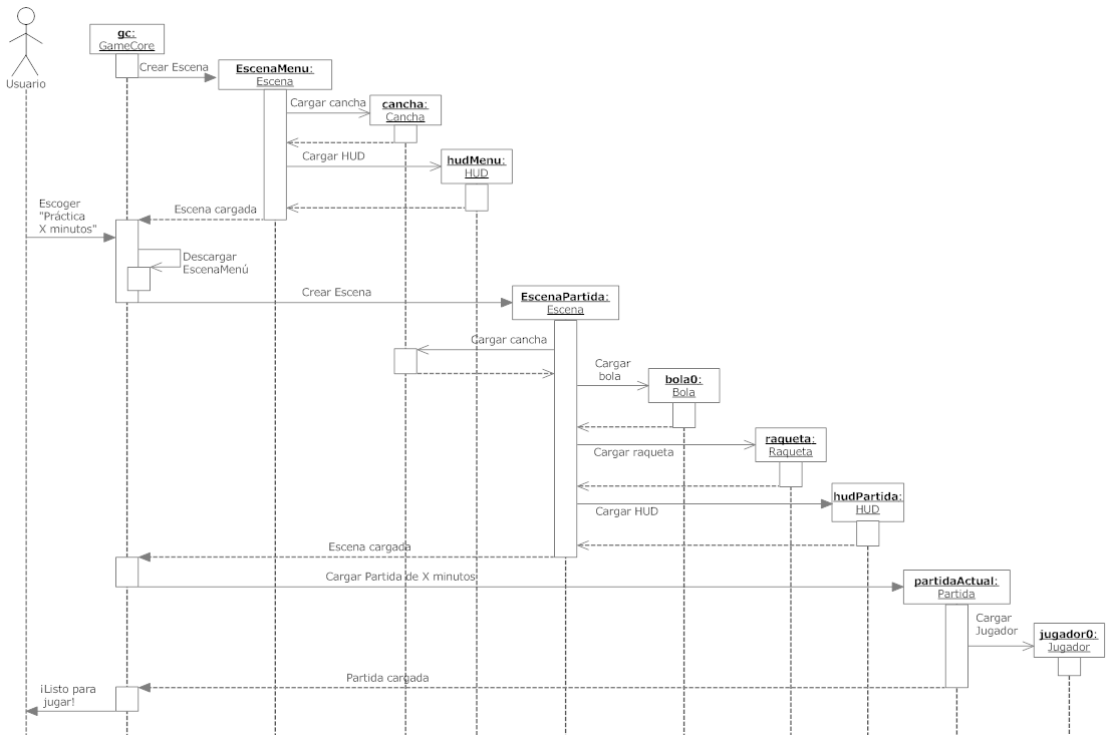


Fig. 18: Diagrama de interacción de objetos del caso de uso 1

Objetos involucrados

- ***gc: GameCore***

  Núcleo del juego. Inicializa objetos y variables globales. Mantiene corriendo el lazo principal, donde se hacen las comprobaciones de estado, se llaman a los callbacks de los objetos y se dibuja cada cuadro. Al salir del lazo, el programa finaliza.

- ***escenaMenú: Escena***

Contiene los objetos que componen una escena del juego, en este caso la del menú, que consiste en la cancha en el fondo, y el HUD en el frente, formando el menú principal del juego.

- ***cancha: Cancha***

  Los objetos que forman la cancha son 6 planos, de los cuales el superior y el de atrás son semitransparentes. Esto último para que se pueda ver la cancha al comenzar a jugar. Se carga una copia completa e igual de la cancha en cada Escena.

- ***hudMenu: HUD***

  Interfaz del menú principal. Incluye el logo del juego y los botones para las opciones: Partida de 1 min, Partida de 3 min, Ver mejores puntajes, Demo y Salir. Contiene las funciones para indicar al cuando se escogió alguna de las opciones.

- ***escenaPartida: Escena***

  Contiene objetos que componen la escena que aparece al cargar la partida. Consiste en la cancha, la bola, la raqueta y el HUD del juego.

- ***bola0: Bola***

  La bola del juego. Aparece sólo en la escena de la partida. Realiza constantes actualizaciones de su estado, comprobando si ha chocado contra la raqueta o alguna de las paredes. Es afectada continuamente por la gravedad de la escena. Entre las propiedades que tiene están la posición, velocidad, aceleración, masa y radio, además de un coeficiente de restitución al chocar contra la raqueta, y otro al chocar contra algún plano de la cancha. Su trayectoria sólo es afectada por la raqueta o al rebotar contra la cancha.

- ***raqueta: Raqueta***

  La raqueta del juego. Aparece sólo en la escena de la partida. Su posición y orientación depende de los datos dados por el sensor Polhemus. En base a esto es calculada su velocidad y aceleración. Dependiendo de estos valores de la raqueta, y de la trayectoria, velocidad y aceleración de la bola, es que calcula hacia dónde va la bola luego del impacto con la raqueta.

- ***hudPartida: HUD***

La interfaz mostrada durante la partida: el puntaje, el contador de tiempo y la barra de mensajes, con sus recuadros. Recibe el estado de la partida para mostrar el respectivo mensaje en la barra, o modificar el puntaje mostrado.

- ***partidaActual: Partida***

  Este objeto incluye el estado y tiempo límite de la partida. Creado al iniciarla, eliminado al finalizarla.

- ***jugador0: Jugador***

  Aquí se mantiene el estado y puntaje del jugador actual, y de ese estado depende si la partida avanza, pues lleva el control de cuántas veces ha fallado el saque inicial. Es creado y asignado a la partida.

Caso de Uso 2: Colisión entre la raqueta y la bola

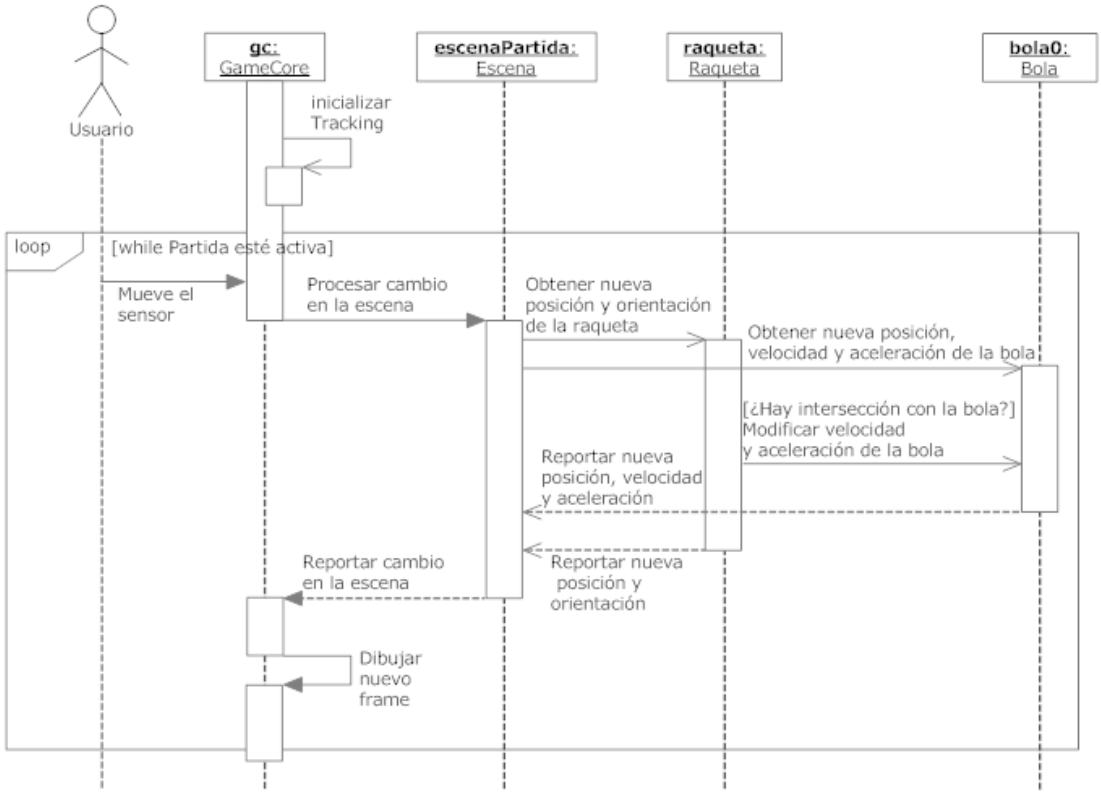


Fig. 19: Diagrama de interacción de objetos del caso de uso 2

Objetos involucrados

- ***gc: GameCore***

  El núcleo del programa. Encargado de inicializar el funcionamiento de los sensores (tracking) y de mantener el lazo de ejecución.

- ***escenaPartida: Escena***

  Encargada de recibir la nueva posición dada por el sensor, y mandársela a la raqueta para que la procese. También reporta de

vuelta al núcleo la totalidad de cambios en la escena, luego de recibir esa información de la bola y la raqueta.

- ***raqueta: Raqueta***

  Recibe la nueva posición y orientación dadas por el núcleo, y reporta el cambio a escenaPartida. Además comprueba si se intersecta con la bola al procesar dichos datos. Si hay intersección, se modifican las nuevas velocidad y aceleración de la bola para que se incluyan en el redibujo de la escenaPartida.

- ***bola0: Bola***

  En cada iteración procesa su nueva posición, velocidad y aceleración, afectadas por los valores de iteración anterior, más la gravedad en la escena, más la velocidad y aceleración dadas por una colisión con la raqueta, en caso de suceder.

Caso de Uso 3: Colisión entre la bola y la pared

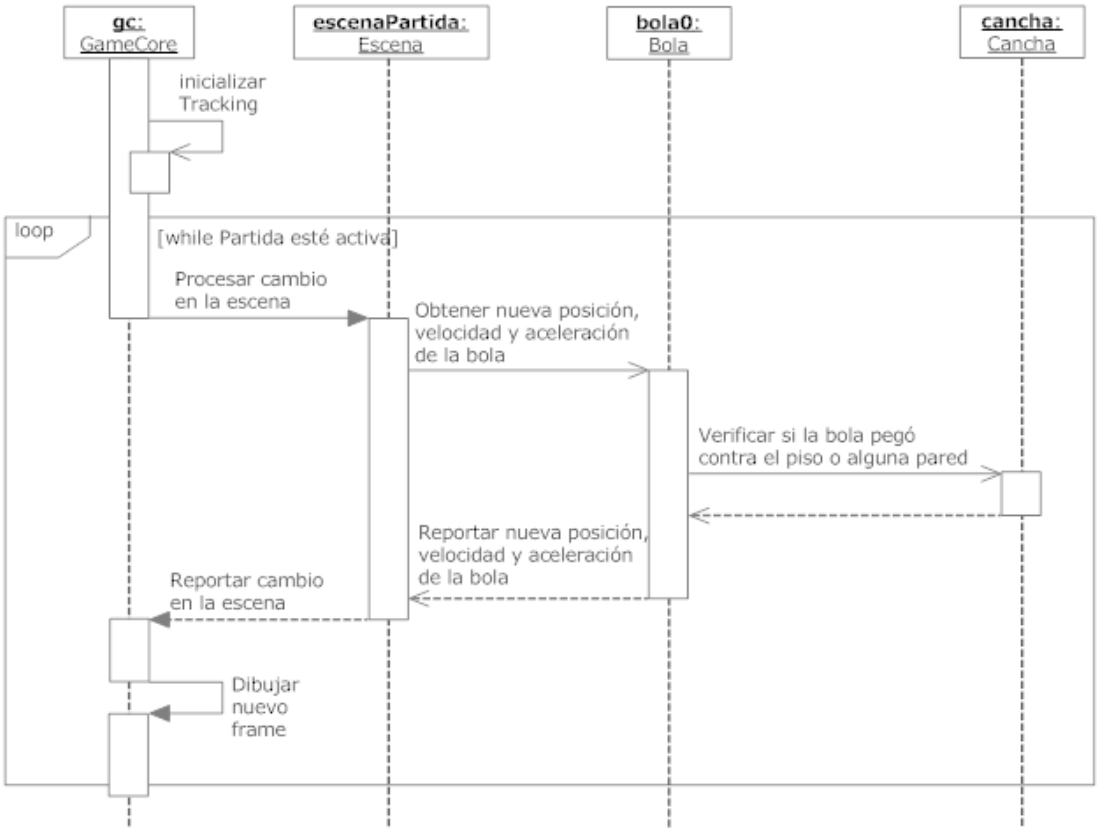


Fig. 20: Diagrama de interacción de objetos del caso de uso 3

Objetos involucrados

- ***gc: GameCore***

  El núcleo del programa. Encargado de inicializar el funcionamiento de los sensores (tracking) y de mantener el lazo de ejecución.

- ***escenaPartida: Escena***

Encargada de pedir y obtener los cambios en las propiedades de la bola, sea que impacte o no contra el piso o alguna pared.

- ***bola0: Bola***

  En cada iteración procesa su nueva posición, velocidad y aceleración, afectadas por los valores de iteración anterior y la gravedad en la escena. También verifica continuamente si ha colisionado contra la cancha. Los valores en sus propiedades cambian acordemente para reflejar si ha colisionado contra una pared o el piso, en caso de suceder.

- ***cancha: Cancha***

  Compuesta de 6 planos, son constantemente consultados por la bola para comprobar si hay colisión con alguno de ellos. Esto para que la bola cambie correctamente su trayectoria y velocidad.

Caso de Uso 4: Ver puntajes altos

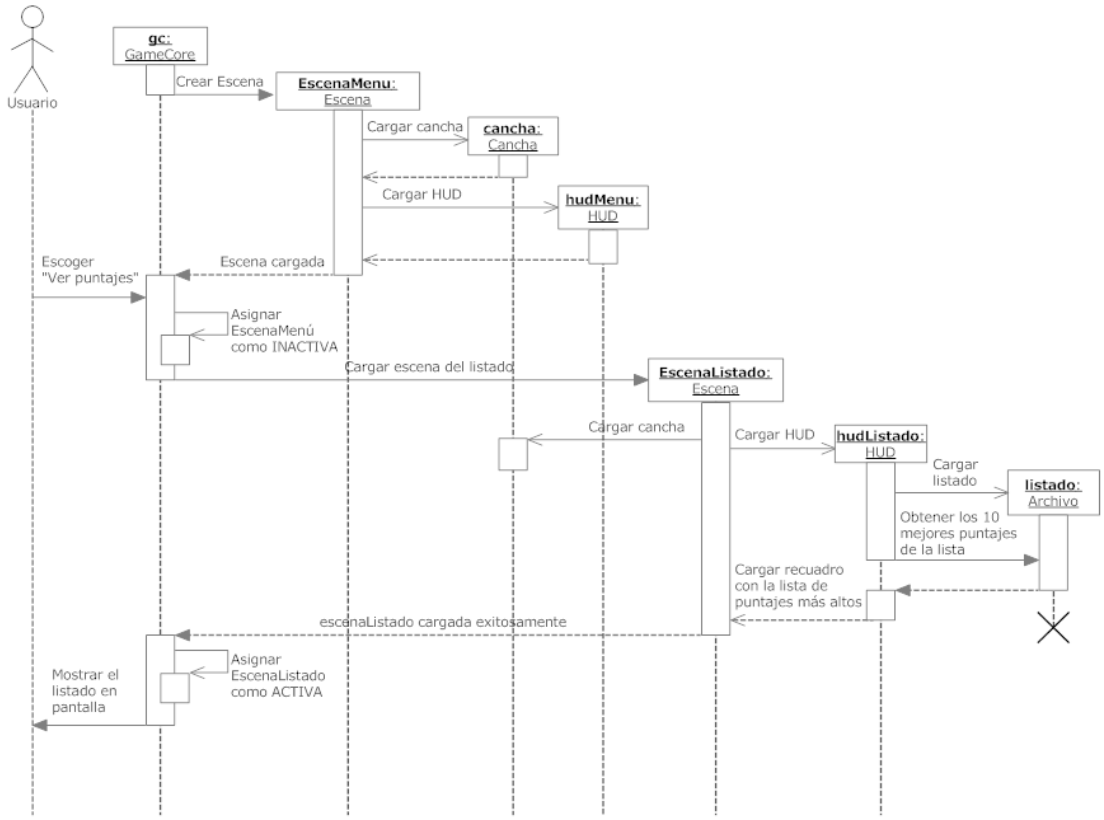


Fig. 21: Diagrama de interacción de objetos del caso de uso 4

Objetos involucrados

- ***gc: GameCore***

  El núcleo del programa. Encargado de inicializar las escenas y de mantener el lazo de ejecución.

- ***escenaMenú: Escena***

Carga y contiene los objetos que componen una escena del juego, en este caso la del menú, que consiste en la cancha en el fondo, y el HUD en el frente, formando el menú principal del juego.

- ***cancha: Cancha***

  Los objetos que forman la cancha son 6 planos. Para usarla de fondo con fines decorativos, tanto en escenaMenu como en escenaListado. Se carga una copia independiente en cada escena.

- ***hudMenu: HUD***

  Interfaz del menú principal. Incluye el logo del juego y los botones para las opciones: Partida de 1 min, Partida de 3 min, Ver mejores puntajes, Demo y Salir. Contiene las funciones para indicar al cuando se escogió alguna de las opciones.

- ***escenaListado: Escena***

  Carga y contiene los objetos que componen la escena donde se muestra la lista de los mejores puntajes. Esos objetos son cancha y HUDListado, donde se despliega la lista en un recuadro.

- ***hudListado: HUD***

  Interfaz de la escena del listado. Es encargado de llamar al objeto listado, y obtener de este el contenido a desplegar. Despliega en un recuadro centrado en pantalla la lista de los mejores puntajes del juego.

- ***listado: Archivo***

  Encargado de abrir el archivo que contiene los nombres y puntajes obtenidos en el juego. Incluye capacidad de procesar la lista y devolver sólo los mejores puntajes.

CAPÍTULO V
IMPLEMENTACIÓN

5.1. Detalles de la implementación

5.1.1. Desarrollo de la aplicación

En la aplicación usamos ciertas ecuaciones, algoritmos de simulación física para el cálculo de: la velocidad y aceleración de la bola, el rebote de la misma en el piso y paredes, el golpe en la raqueta, la velocidad y aceleración de la raqueta, y por último la gravedad que influye sólo a la bola.

La bola tiene definidos diversos parámetros: masa, posición, velocidad y aceleración. La raqueta tiene los mismos, más la orientación. Los únicos datos de entrada son la posición y orientación de la raqueta, obtenidos del sensor de movimiento.

Todos los parámetros son calculados en cada cuadro de la animación, y siempre se tiene en cuenta el rango de cada cuadro actual, pues se necesita la duración de cada cuadro para el correcto cálculo de los parámetros.

Escenario

Creación del escenario y los objetos usando 3D Studio Max

Para crear los escenarios del juego fue necesario saber las medidas exactas de la cancha del Squash, de qué color normalmente son las paredes, materiales, líneas, cuadros, etc. También la distancia entre donde está el jugador la pared en donde choca la bola.

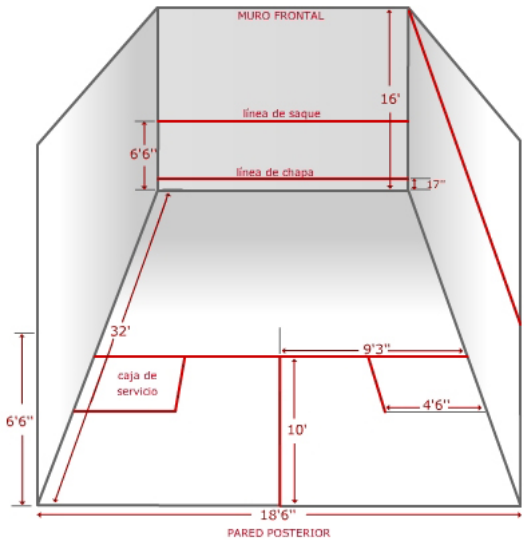


Fig. 22: Medidas de la cancha de Squash

El modelo se lo escalo a cada 10 pixeles = 1 cm. Usamos imágenes de textura de madera y el símbolo de la ESPOL para el piso y las paredes de color gris con las rayas rojas.

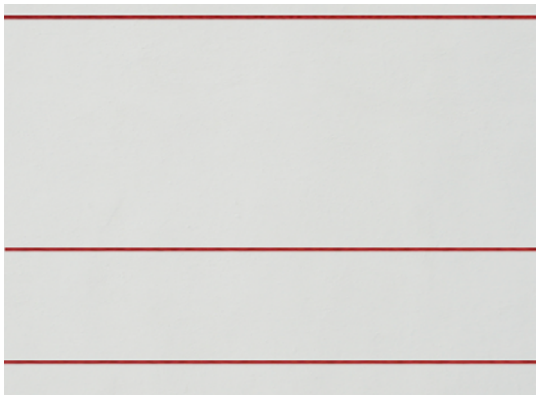


Fig. 23: Pared de fondo de nuestra cancha de squash

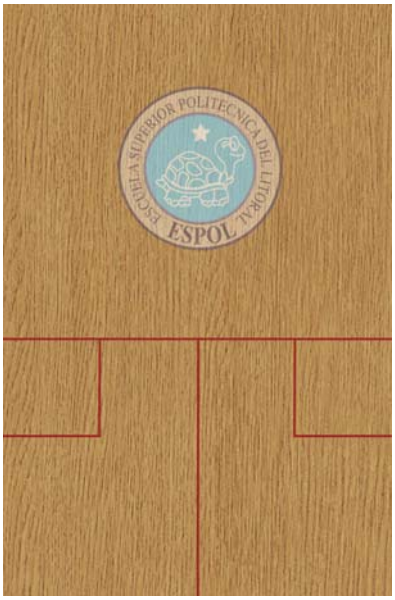


Fig. 24: Piso de nuestra cancha de squash

La raqueta de igual forma fue modelada en 3D Studio Max™ y de igual forma fue escalada para que pueda entrar en la cancha.

Por último la pelota fue solo una esfera creada dentro de OpenSceneGraph de color negro, ya que es la bola normal que se juega dentro del juego Squash.

Sonido de la aplicación

Consiste en un par de pistas musicales, y efectos de sonido para las diversas acciones en el juego. Utilizamos la librería SDL, junto con la librería SDL Mixer 1.2, para fácilmente controlar los sonidos. Todos ellos dados como archivos OGG o WAV, gracias a que son soportados directamente por la librería, sin necesidad de plugins adicionales.

Bola

Movimiento de la bola

El movimiento de ella siempre está influenciado por la gravedad, simulada como una aceleración que siempre apunta hacia abajo. Esto para evitar que la bola rebote sin parar hacia todos lados, como si estuviera en el espacio. De ahí, el movimiento de ésta sólo es alterado cuando golpea contra una pared o el piso, o cuando le pega la raqueta, detallados en las secciones siguientes respectivamente.

Raqueta

Movimiento de la raqueta

A la raqueta modelada en el juego se le tiene definido dos puntos principales: el pivote y el centro de la malla.

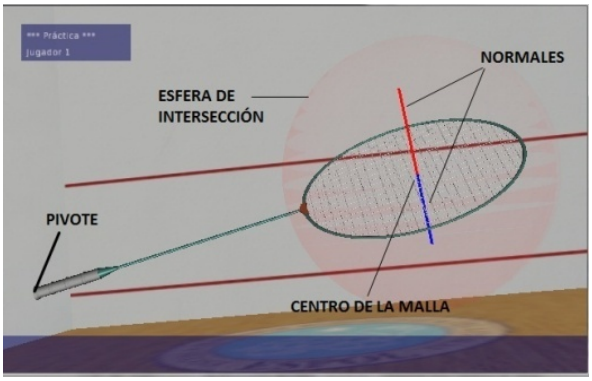


Fig. 25: Componentes de la raqueta en el juego

El pivote está ubicado en la punta del mango, y es el punto de referencia para el movimiento y orientación de la raqueta. Es decir, la posición dada por el sensor de movimiento está representada en el ambiente virtual por este punto. Lo mismo para la orientación, la cual es alrededor de este mismo punto.

El centro de la malla está, como dice, en el centro de la malla donde la bola debe golpear. Se usa como el centro de una esfera virtual, que

contiene a toda la malla, y es usada para el cálculo de la intersección entre la bola y la raqueta, descrito con anterioridad en el análisis. También se lo usa para dibujar las normales de los planos virtuales, usados para el rebote de la bola con la raqueta. Tanto las normales como la esfera virtual son visibles sólo con fines de prueba. Éstas no son visibles en la versión final.

Rebotes

Dado el análisis procedimos a implementar los rebotes contra las paredes y pisos de acuerdo a la física explicada con anterioridad, esto nos permite tener rebotes más realistas.

5.1.2. Versiones del proyecto

Existieron varias versiones del proyecto lo cual nos ayudaban a entender la mayoría de errores que se tenían en el juego. El Dr. Sixto García nos enfocó siempre la idea de que el juego debe ser lo más parecido a la realidad, lo cual al principio no era nuestro objetivo principal, ya que obtuvimos gran cantidad de escenarios antes de llegar a aceptar esta idea.

Versión 1.0

La primera idea que obtuvimos del juego fue la que jugaríamos en el espacio, sin tener un campo de gravedad definido, teniendo que la cancha estaba amarrada a dos naves espaciales y que se iba moviendo por el espacio, lo cual nos dimos cuenta que no demostrábamos en si el juego Squash, sino un juego completamente distinto.

También usábamos otro dispositivo que fuimos dejándolos a un lado ya que no cumplían con la expectativa de idear un juego basado en la realidad, este dispositivo fue el Wii-Mote™ que al principio nos pareció un dispositivo necesario para el juego, ya que su conexión bluetooth era necesaria para adaptarlo al Squash, pero los movimientos eran completamente irreales, ya que la librería que usábamos no estaba completamente diseñada para movimientos tan bruscos como las de una raqueta, además la librería era BETA, y no nos proporcionaba la información necesaria del Wii-Mote™. Esta librería es de código abierto, pero la triangulación del movimiento era muy complicada y por el tiempo que tuvimos no nos fue tan fácil crear las funciones que necesitábamos para que el movimiento sea lo más real posible.

Fue por esto que decidimos dejar la idea del espacio y del uso del Wii-Mote™.

Fig. 26: Wiimote, dispositivo usado en el primer prototipo

Fig. 27: Adaptador Bluetooth USB, dispositivo usado en el primer prototipo

Luego comenzamos con la primera versión del juego que se viera más seria y formal y para comenzar desarrollar una física que nos permitiera rebotes de la bola de forma real.

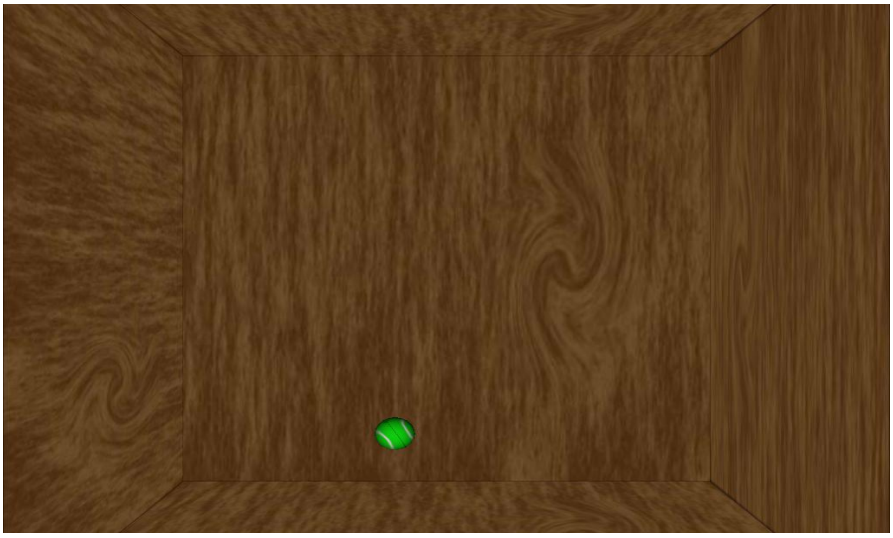


Fig. 28: Primer prototipo del juego Squash

Versión 2.0

El proyecto en sí, tenía muchas fallas por lo que es la física del mismo, la velocidad de la gravedad, la masa de la bola, la masa de la raqueta, la sensibilidad del sensor, el tamaño de la pantalla, la escala de movimientos, la distancia entre el emisor y el usuario, y muchos problemas, que se fueron remediando, con cada pruebas respectiva.

La gravedad en esta evolución fue de 9.8 m/s^2, nos mostraba el movimiento de la caída real de la bola, pero era sumamente complicado golpearla de nuevo por la velocidad que ejercía hacia abajo, por lo que decidimos dejar la gravedad a 6.5 m/s^2, para que la

bola caiga más lento y se pueda pegar con la raqueta mucho más fácil, y hacer el juego más entretenido.

La sensibilidad del sensor, era otro inconveniente muy importante, ya que el movimiento de la raqueta debía ser lo más real posible, por lo que nuestro primer intento fue con la sensibilidad a 1:1 (completamente proporcional), de igual forma que la gravedad tuvimos problemas porque los movimientos eran muy rápidos y la bola pasaba de un lado al otro y de cada 10 golpes solo 1 a 2 tocábamos la bola, por lo que decidimos probar dejando la sensibilidad multiplicada a 1.5 por lo que dejo que la raqueta sea más precisa y de cada 10 golpes lanzados tocábamos la bola 6 a 7 veces, por lo que el juego se volvía un poco más fácil.

El tamaño de la pantalla con la escala de movimientos, era problemas que tenían mucho en común, ya que según esto debíamos proporcionar la escala para la cancha de Squash, dejando que por cada 10:1 (cm) en la cancha.

Fig. 29: Pantalla inicial del juego, segundo prototipo.

En esta versión también comenzamos a definir cuáles son las opciones que nos iba a ofrecer las opciones principales del menú, que en este caso habíamos pensado en un tiempo para practica y un tiempo más prolongado para una partida simple.

Versión 3.0

Para terminar el proyecto se tuvieron que hacer muchas pruebas para determinar una velocidad más apropiada donde el jugador pudiera interactuar de manera sencilla con la bola, también hubieron ajustes de cámara y de sombras para mejorar la apreciación de la percepción dentro de la cancha de juego. A continuación se muestra las opciones

que definimos para el menú principal y como luce la pantalla de una partida.

Menú principal

Al iniciar el juego, el menú principal es lo que nos aparece. Sólo muestra las opciones para “Práctica 1 minuto”, “Práctica 3 minutos”, “Mejores Puntajes” y “Salir”. Se usa el ratón para seleccionar la deseada. El juego en sí es con el sensor de movimiento.

Fig. 30: Pantalla inicial del juego

Opciones del menú principal

Existen 4 opciones del menú principal, las cuales las dos primeras se refieren al juego en sí, que es practicar el squash 1 minuto o 3 minutos, para sacar un puntaje respectivo, según el tiempo determinado, el tercer botón se refiere a mostrar en pantalla los puntajes más altos de aciertos que tenga un jugador, presentando una pantalla como esta:

Fig. 31: Mejores puntajes, mostrados en la aplicación

Y por último se encuentra el botón de salir que lo que hace es terminar el programa y salir del mismo, liberando todos los espacios de memoria usados.

Mejores puntajes

Para estar en los diez primeros puntajes, fue necesario crear un algoritmo de orden ascendente sacando el mayor número de aciertos, los cuales se los toma al final de la práctica, con el nombre del jugador y se los graba en un archivo de texto, lo cual al poner la opción de "mejores puntajes" los muestra de forma ordenada. Al final de la práctica aparece esta pantalla, la cual muestra el número de aciertos y pide el nombre del jugador:

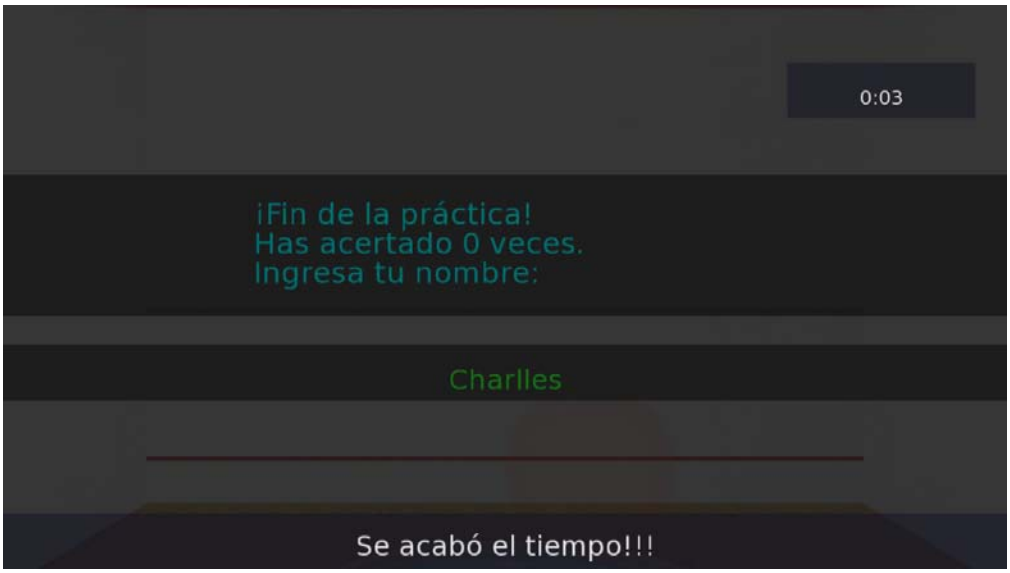


Fig. 32: Pantalla al final de la práctica, donde se pide el nombre al jugador

Práctica de uno o tres minutos

Este es el juego en sí, cumpliendo las reglas correspondientes del juego Squash, comenzando con la primera regla que es en el saque pegarle en medio de las dos rayas rojas de la pared de adelante, y después de esto golpear la bola sin que pase la raya roja que se encuentra en la parte de arriba tanto de las pared frontal como de las paredes laterales. Por cada vez que se la bola haya sido correctamente golpeada se le suma un punto en los aciertos, y si no se suma un punto en las fallas.

En el otro extremo de la pantalla se muestra el tiempo que transcurre a medida que se esté jugando, cuando se acaba el tiempo, aparece una pantalla indicándolo, lo cual hace que el juego se detenga y aparezca la pantalla para escribir el nombre del jugador:

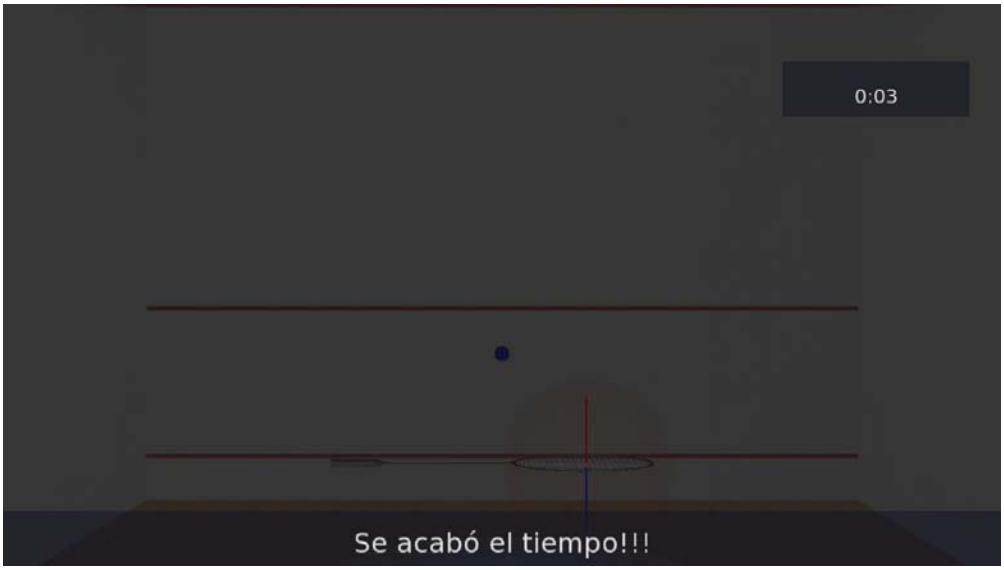


Fig. 33: Pantalla que aparece al acabarse el tiempo de la práctica

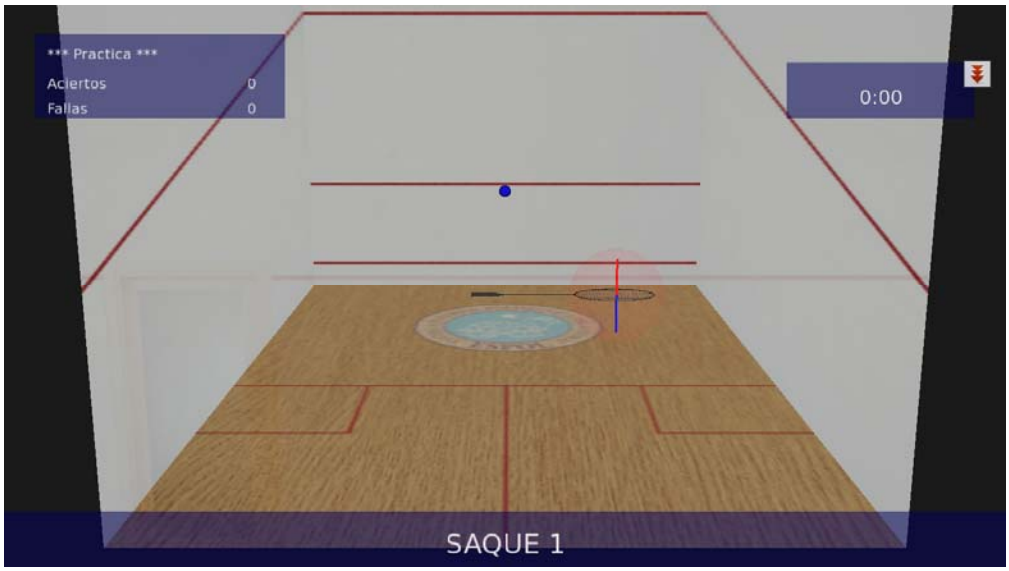


Fig. 34: La cancha de Squash

5.1.3. Pruebas

Las pruebas nos ayudaron a poder obtener un producto final con el que el usuario pueda interactuar de tal manera que esté a gusto con el juego de "squash", Las cuales son mencionadas a continuación:

Primera Prueba: Conexión de equipos

En la conexión de los equipos nos tomó cierto tiempo encontrar las fallas de los drivers de los mismos, o las inclusiones dentro del proyecto en sí, variables de entorno, etc. Que a la larga después de una carta a los creadores de estos aparatos nos pudimos conectar correctamente. Para las pruebas del proyecto utilizamos los diversos dispositivos de realidad virtual, como son los sensores, el casco HMD, el guante, etc. Lo que nos permitió darnos cuenta cuales eran los ideales para nuestro tipo de proyecto los cuales son un sensor pegado a un Wii-mote, las gafas estereoscópicas con su respectivo sensor de movimiento, y el proyector Depth a 120Hz de frecuencia. Y los conectamos como se encuentra en las siguientes figuras:

Fig. 35: Conexión del proyector y de las gafas estereoscópicas

Fig. 36: Gafas estereoscópicas con su sensor

Fig. 37: Wii-Mote con el sensor de movimiento

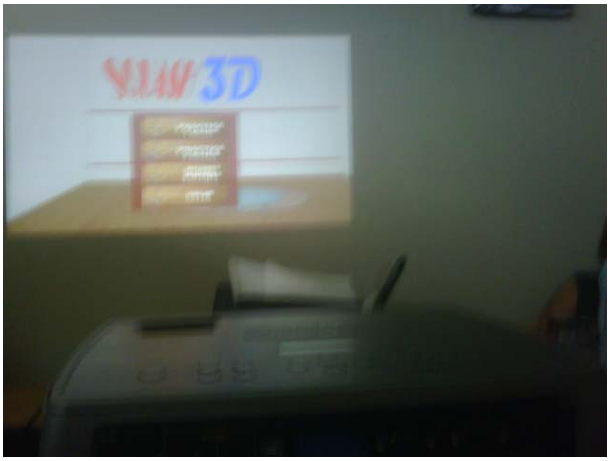


Fig. 38: Proyector DepthQ trabajando

Segunda Prueba: Cámara

La segunda prueba fue la posición de la cámara en el juego y el seguimiento de la misma. La primera opción que tuvimos fue de que la cámara siga a la bola, hasta en los rebotes, pero solo de forma 2D, o sea sin profundidad, solo se movía horizontal y verticalmente, como lo observamos en la siguiente figura:

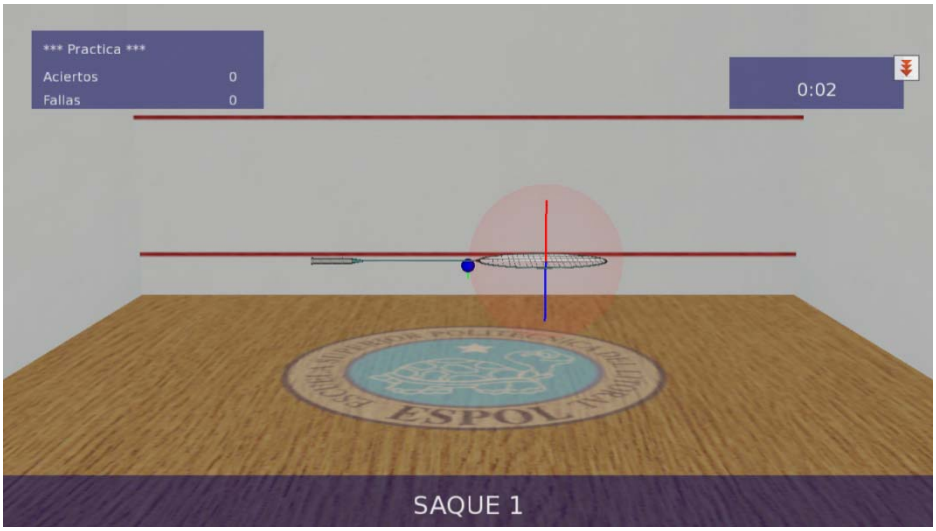


Fig. 39: Posición de la cámara siguiendo la bola

Pero nos dimos cuenta que al jugar, la persona se iba a marear muy fácilmente, las pruebas de visión nos dieron como resultado que no era lo correcto este movimiento de cámara. Por este motivo fue que recordando el juego de tenis en el juego Wii-Sports nos dimos cuenta que la mejor posición de la cámara es en el extremo superior de la intersección de la pared de atrás con el techo, y colocamos tanto estas paredes transparentes, así como se muestra en la figura:

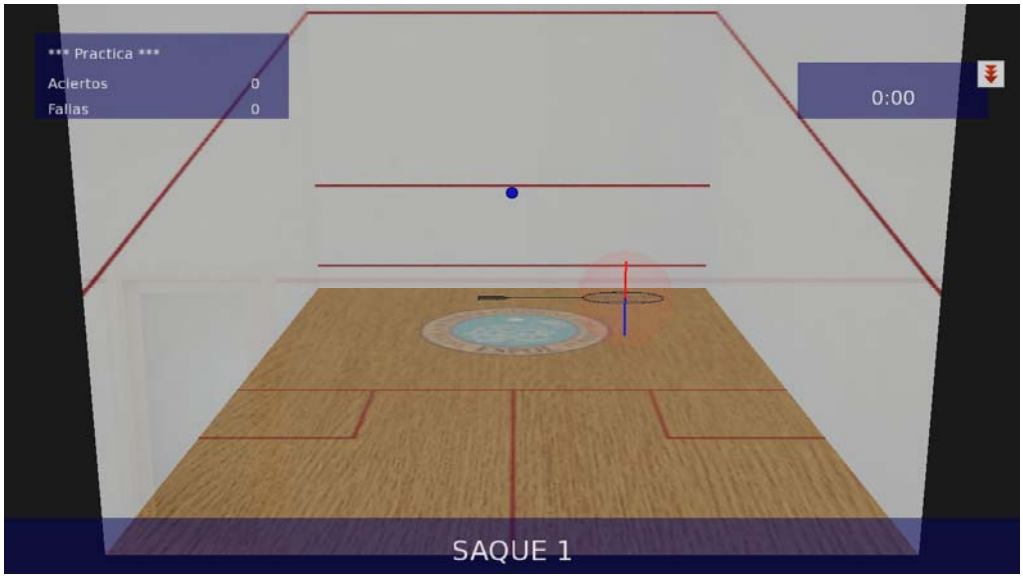


Fig. 40: Posición final de la cámara nos permite percibir profundidad y observar toda la cancha

Tercera Prueba: Escalamiento a causa del movimiento

La tercera prueba fue acerca de posición de la raqueta. El problema era que al tener un área limitada por los sensores, ya que estos no son inalámbricos, no podíamos recorrer el espacio que ocupaba la cancha virtual.

La solución se dio al escalar el movimiento para lo cual cambiamos el movimiento normal tomado por los sensores y los escalamos de 1 a 1.5 lo cual nos permitió que al mover la raqueta esta recorriera más espacio dentro del juego.

Cuarta Prueba: Velocidad de la pelota

La cuarta prueba fue acerca de la velocidad que tiene una pelota de "squash". El problema es que la pelota con un movimiento real es bastante rápido por lo que hay que tener entrenamiento para poder jugar de forma agradable.

La solución para hacer que el juego sea disfrutado por usuarios que no hayan practicado un deporte similar es reducir la velocidad con que la pelota se movía dentro del escenario. La gravedad original hacía que la bola bote muy rápido. Por lo que optamos por una reducción de la gravedad simulada en la aplicación de 9.8 m/s^2 a 6.5 m/s^2, la cual se obtuvo luego de realizar un muestreo de la velocidad que la bola tiene en distintos tiempos, al quedarse botando en su sitio. La aceleración finalmente escogida fue la que mostró una menor velocidad promedio en los respectivos tiempos, pero sin ser demasiado baja, pues hacía que el movimiento fuese lento e irreal.

		Velocidad (m/s)			
	Aceleración (m/s^2)	9,80	7,00	6,00	6,50
Tiempo (s)	1	709,80	623,74	428,28	407,31
	2	69,29	10,65	185,66	234,00
	3	338,13	466,25	286,33	301,82
	4	619,08	229,28	313,67	327,58
	5	341,17	73,18	53,47	31,46
	6	256,45	283,16	306,30	197,30
	7	354,75	400,17	280,41	263,72
	8	134,16	303,71	144,11	300,95
	9	237,59	313,98	72,63	80,66
	10	652,28	177,17	83,94	96,93
	Velocidad promedio	**371,27**	**288,13**	**215,48**	**224,17**

Tabla 1: Muestreo de velocidades de la bola en diferentes tiempos

Quinta Prueba: Colisión de la bola con la raqueta

La quinta prueba fue una falla que había al colisionar la raqueta con la bola. El problema se daba cuando la tasa de cuadros por segundos era baja y cuando trataba de calcular la colisión solía pasar que en un cuadro era muy temprano para hacerlo y en el siguiente ya era muy tarde por lo que la bola atravesaba la raqueta.

La solución a este problema vino junto con la solución de otro, este era que al tener la cámara en la posición actual no se podía percibir correctamente la raqueta por lo cual se decidió escalarla con lo que el área donde se calculaba las colisiones también se agrando, además

tuvimos que volver a la escala de movimiento normal al tener la raqueta de mayor tamaño abarcábamos más área al golpear y el uso del proyector y las computadoras del laboratorio mejoraron la tasa de cuadros por segundos y se solucionó el problema.

Sexta Prueba: Percepción de distancia entre raqueta y bola

La sexta prueba se dio ya que es verdad que se podía percibir mejor la profundidad de la cancha, pero aún era muy difícil apreciar las pequeñas distancias entre la raqueta y la bola.

Esto se solucionó con el implemento de luces y sombra. Al principio descubrimos que la librería de OSG nos brindaba una forma de colocar haces de luz focalizada. Utilizando un plano compuesto por una matriz de vértices, y no por sólo las 4 esquinas como era inicialmente, la luz daba un aspecto más realista a las paredes y piso. Pero la sombra no aparecía sobre los planos. Así que decidimos utilizar otra alternativa, donde en lugar de planos utilizábamos cubos aplastados, basándonos en uno de los tutoriales de OSG. En ellos sí se proyectaban sombras. De esta manera obtuvimos la sombra de la bola y la raqueta

que nos indican que tan separados están y con una fuente de luz que ilumina toda la cancha.

CONCLUSIONES Y RECOMENDACIONES

Nuestras conclusiones son:

1. Todo parece indicar que la evolución de sistemas de Realidad Virtual, se esforzará en permitir que la participación principal del usuario no especializado este se centrara en el desarrollo y aplicación de habilidades y de conocimientos orientados a la concepción, diseño y construcción de mundos virtuales.

2. Pudimos desarrollar un juego de práctica de Squash, con los dispositivos deseados, y la meta deseada. Recreando un escenario muy parecido a la realidad, con las medidas justas, la gravedad necesaria y el movimiento captado por los trackers, escalado de manera exacta.

3. Pudimos apreciar las dificultades que aparecen al desarrollar una aplicación utilizando este tipo de tecnologías. Es el caso de recrear leyes de la física necesarias para simular lo más fielmente posible a la realidad. Y aunque se apliquen dichas leyes físicas conocidas, sucede que el

comportamiento de los objetos no se observa de forma tan natural, por lo cual uno debe de realizar ajustes respectivos.

4. Otra dificultad fue encontrar la posición adecuada de la cámara dentro del entrono tridimensional. Esto para poder observar el juego con la perspectiva correcta. Nos dimos cuenta que aunque los dispositivos te dan la facilidad de poder apreciar en tres dimensiones la escena del juego, sin una ubicación correcta de la cámara, este efecto no es nada o apenas percibido, y pierde toda utilidad.

5. A diferencia de los objetos reales, los objetos modelados se traslapan y cruzan. Así que para el uso de esta tecnología, todo comportamiento, desde el más sencillo, como la colisión de dos cuerpos, debe ser programado como tal.

6. Se debe tener también amplio conocimiento matemático acerca de las técnicas utilizadas en gráficos por computadora, ya que se deben crear reflejos y sombras por nuestra cuenta para lo cual ya existen algoritmos.

7. Lo más importante de la tecnología de Realidad Virtual es la inmersión que se obtiene. Ésta debe envolver a la persona para poder jugar con nuestros sentidos, y comenzar a pensar que lo que estamos haciendo o en nuestro caso jugando es casi real.

Y nuestras recomendaciones son:

1. Se recomienda que al momento de usar el programa de Squash, el emisor deba estar a una altura de 1 metro del suelo y en la parte de atrás del jugador. Recordemos que el emisor sólo envía señales en un radio omnidireccional de 1.5 metros, por ende no se puede ponerlo tan lejos de la persona. Sería mejor usar otro tipo de emisor más potente, para reacomodar la escala de la cancha y poder jugar casi de forma real.

2. Para lograr un mayor grado de inmersión se debería completar el desarrollo con el proyector. Sería implementar una "caverna" para que así las imágenes nos rodeen y no sólo verla de frente. Otra opción para la inmersión sería la utilización del casco HMD. Con este podríamos seguir el movimiento de la bola simplemente al mover la cabeza. Estas opciones

remplazarían a la cámara estacionaria, y nos mantendría dentro del área de la cancha.

3. Sugerimos seguir ampliando el juego, ya que hemos visto que este tipo de tecnologías es muy aplicable a esta clase de proyectos.

4. Podríamos extender aún más la aplicación, dedicándola al estudio de los movimientos de los jugadores de este deporte. El objetivo es poder estudiar sus rutinas de entrenamiento, encontrando errores y ayudar a mejorar su rendimiento en la cancha.

5. Nuestro juego consiste en sólo prácticas de un jugador, ya que nuestro objetivo principal era el uso de dispositivos para la realidad virtual en un sistema multimedia, y no el juego de dos jugadores. Pero dentro de nuestro código sí existen ciertos parámetros que servirán de ayuda para desarrollar el Squash para 2 jugadores. Sería muy interesante poder seguir con la investigación y creación, dentro del mercado ecuatoriano, de estas técnicas y juegos.

ANEXOS

ANEXO A

ENCUESTA SOBRE REALIDAD VIRTUAL

Pregunta 1. ¿Qué edad tiene? (si es mayor de edad) ¿Tiene hijos? (si no tiene hijos, no es necesario que haga la encuesta).

_____

Pregunta 2. ¿Usted conoce la tecnología de realidad virtual?

Sí ____ No____

Pregunta 3. Después de observar las imágenes anaglíficas, y tuviera la oportunidad de utilizar esta tecnología ¿La usaría?

Sí ____ No____

Pregunta 4. En el área del entretenimiento, ¿usted estaría dispuesto a experimentar una propuesta con esta tecnología?

Sí ____ No____

Pregunta 5. ¿Estaría dispuesto a invertir en un juego que pudiese compartir con sus hijos?

Sí ____ No____

Pregunta 6. ¿Estaría dispuesto a jugar Squash en la sala de su casa utilizando realidad virtual?

Sí ____ No____

Pregunta 7. Y si es así; ¿Cuánto usted estaría dispuesto a pagar por una entrada a un sitio que le permita experimentar un juego así?

($) 1 – 2.99____ ($) 3 – 4.99 ____ ($) 5 – 10____ ($) más de 10___

Pregunta 8. ¿En qué áreas usted cree que también le gustaría que se use la realidad virtual?

Medicina _____ Juegos_____ Spa_____ Turismo_____

Películas_____ Otros (Especifique) _____

Pregunta 9. ¿Cree usted que la realidad virtual mejoraría el realismo y la inmersión de los juegos? (ponga del 1 al 5, 5 siendo la puntación máxima).

1___ 2___ 3___ 4___ 5___

ANEXO B

DESCRIPCIÓN DE MÓDULOS

Modulo Bola

Nombre: `Bola::Bola( osg::Vec3 centro, float radio, std::string nombre)`

Descripción: *Método que retorna una bola con todos sus atributos.*

Tipo dato retorno: *Retorna un tipo de dato bola que consta de de posición centro, radio textura y ubicada dentro del árbol, además de mostrar la trayectoria del desplazamiento en el debug.*

Tipo dato entrada: *recibe un* `osg::Vec3 para el centro, un float para el radio, y un string para el nombre`.

Nombre: `void Bola::setPosition (const osg::Vec3 &newPos)`

Descripción: *Método que cambia en el árbol la posición de la bola en el nodo de desplazamiento.*

Tipo dato retorno: *retorna* `void` *pero cambia un nodo del árbol.*

Tipo dato entrada: *recibe un* `osg::Vec3` *con la nueva posición*.

Nombre: `void Bola::setOrientation (float rad1, osg::Vec3 &eje1, float rad2, osg::Vec3 &eje2, float rad3, osg::Vec3 &eje3)`

Descripción: *Método que setea la matriz* `_patPos`.

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe los ángulos y sus ejes respectivos.*

Nombre: `void Bola::setOrientation (osg::Quat &newOrientQuat)`

Descripción: *Método que setea la matriz* `_patPos`.

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada*: recibe un* `osg::Quat de la nueva orientación`.

Nombre: `void Bola::setColor (osg::Vec4& color)`

Descripción: *setea el color de la bola.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe un* `osg::Vec4` *con el color.*

Nombre: `void Bola::setMasa(float masa)`

Descripción: *setea la masa de la bola.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe un* `float` *con el valor de la masa.*

Nombre: `void Bola::setVolumen(float vol)`

Descripción: *setea el volumen de la bola.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe un* `float` *con el valor del volumen de la bola .*

Nombre: `void Bola::setDensidad(float dens)`

Descripción: *setea la densidad de la bola.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe un* `float` *con el valor de la densidad de la bola.*

Nombre: `void Bola::setCallback(osg::NodeCallback* nc)`

Descripción: *es un callback para actualizar la matriz* `_patPos`.

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *es un callback.*

Nombre: `void Bola::agregarALaEscena(osg::Group* grupo)`

Descripción: *agregar un nuevo* `_patPos al osg::Group` *grupo.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *recibe un* `osg::Group de la escena donde se va a agregar`.

Nombre: `void Bola::CalcularColisionConPiso(float coef_restitucion, osg::Vec3 normal)`

Descripción: *calculo del la colision de la bola contra el piso.*

Tipo dato retorno: *no retorna nada pero llena la matriz de las velocidades de cada eje*.

Tipo dato entrada: *recibe un* `float` *y un* `osg::Vec3 correspondiente al coeficiente de restitución de la bola y la velocidad de la normal`.

Nombre: `bool Bola::CalcularColisionConRaqueta(Raqueta::t_PlanoNum planoNum, Raqueta* raqueta, float disBolaRaq)`

Descripción: *calcula la colisión con la raqueta de acuerdo con:*

$$V_{nb2} = \frac{(C_R+1)M_r V_{nr1}+V_{nb1}(M_b-C_R M_r)}{M_b+M_r}$$

Tipo dato retorno: *retorna un booleano que indica si se pudo o no realizar el cálculo*.

Tipo dato entrada: *recibe* `una Raqueta` *además de la distancia de la bola a la raqueta.*

Módulo Callbacks

Los métodos y funciones de esta clase son las que se ejecutan en cada frame.

Nombre: `void printFirst(GameCore *gc, int objCnt)`

Descripción: *Método que nos muestra los settings y el resultado de nuestras elecciones en el juego.*

Tipo dato retorno: *no retorna ningún dato pero imprime en pantalla opciones.*

Tipo dato entrada: *recibe un* `GameCore y un objeto contador .`

Nombre: `void clearScreen()`

Descripción: *función que sirve para limpiar el texto de la pantalla.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `BolaCB::BolaCB( GameCore* gc, Bola* bola)`
Descripción: *función que sirve de callback para la bola.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `void BolaCB::operator()`
Descripción: *Operaciones del callback para la bola y las colisiones con el piso y las paredes.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `RaquetaCB::RaquetaCB( GameCore* gc, Raqueta* rActual )`
Descripción *función que sirve de callback para la raqueta.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `void RaquetaCB::operator()`
Descripción: *Operaciones del callback para la raqueta y las colisiones con la bola, además se obtienen todos los valores requeridos para el cálculo como velocidades y aceleración.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `MenuPickingEH::MenuPickingEH( GameCore* gc )`
Descripción: *Método constructor del menú principal.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe un* `GameCore`.

Nombre: `bool MenuPickingEH::handle( const osgGA::GUIEventAdapter& ea, osgGA::GUIActionAdapter& aa, osg::Object* pObject, osg::NodeVisitor* pNodeVisitor )`

Descripción: *función del manejador de los eventos del menú principal, este maneja el cambio de imágenes y de sonidos de dicho menú y movimientos del mouse.*

Tipo dato retorno: *retorna un booleano de acuerdo a si se puede ejecutar la operación.*

Nombre: `void MenuPickingEH::cursorSobreBoton(bool* estaAdentroDelBoton, HUD_Element* he, float pX, float pY)`

Descripción: *función para el cambio de imagen del cursor que se ejecuta al estar este sobre el botón.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe un booleano que indica si el elemento se encuentra dentro del botón y las coordenadas del cursor.*

Nombre: `CameraCB::CameraCB( GameCore* gc, SENSOR_CABEZA sensorNum )`

Descripción: *callback de la cámara.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe un* `GameCore` *actual y el sensor de movimiento.*

Nombre: `void CameraCB::operator()`

Descripción: *función que de las operaciones que realiza la cámara en el juego.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Módulo Escena

Nombre: `Escena::Escena(t_tipoEscena claseDeEscena, Partida* partidaActual, unsigned int pantallaW, unsigned int pantallaH) : _grupo(new osg::Group())`

Descripción: *procedimiento constructor de una escena.*

Tipo dato retorno*: no retorna nada sirve para crear la escena dentro de él.*

Tipo dato entrada: *recibe un* `t_tipoEscena` *que es el tipo de escena que se desea crear, también se recibe una* `Partida` *que en este caso es la partida actual además de las medidas de la pantalla recibida con dos* `int` *y un* `_grupo` *donde se encuentran nuestros objetos.*

Nombre: `Escena::~Escena()`
Descripción: *procedimiento que sirve de destructor de una escena y limpia el árbol donde se encontraban nuestros objetos*.
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe ninguno.*

Nombre: `void Escena::agregarAlViewer(osgViewer::Viewer& viewer)`
Descripción: *Método público que sirve para agregar una escena al viewer.*
Tipo dato retorno*: no retorna nada pero agrega la escena al viewer*.
Tipo dato entrada: *recibe* `osgViewer::Viewer` *viewer que estamos usando .*

Nombre: `void Escena::HUD_Partida_inicializarHUD(Partida *partidaActual)`
Descripción: *Método público que inicializa el HUD de la pantalla de la partida actual*.
Tipo dato retorno: *no retorna nada*.
Tipo dato entrada: *recibe una* `Partida` *que es la partida actual a la que agregaremos el Hud.*

Nombre: `void Escena::HUD_Partida_inicializarNombresYPuntaje()`
Descripción: *Método que sirve para inicializar el Hud de una partida inicializando los nombres y puntajes respectivamente.*
Tipo dato retorno*: no retorna nada.*
Tipo dato entrada: *no recibe ninguno.*

Nombre: `void Escena::HUD_Partida_cambiarNombresYPuntaje(Partida *partidaActual)`
Descripción: *Método para cambiar el nombre y los puntajes dependiendo de la partida.*

Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe una* `Partida` *que es la que le sirve para comparar si es la misma partida o una diferente.*

Nombre: `void Escena::HUD_Partida_Reset (Partida *partidaActual)`
Descripción: *Método para resetear el Hud dependiendo de la partida.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe una* `Partida` *que es la que le sirve para comparar si es la misma partida o una diferente.*

Nombre: `void Escena::HUD_Partida_QuitarHUD ()`
Descripción: *Método para quitar el Hud del árbol.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `void Escena::crear(t_tipoEscena claseDeEscena)`
Descripción: *Método para crear todos los componentes de una escena dependiendo que tipo de escena sea o el menú, o una partida.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe* `t_tipoEscena` *con la clase de escena a crear.*

Nombre: `void Escena::printDebug(t_tipoEscena clase)`
Descripción: *Método para impresión en la depuración dependiendo que tipo de escena sea o el menú, o una partida.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe* `t_tipoEscena` *con la clase de escena a crear.*

Nombre `void Escena::cargarCanchaSquash()`
Descripción: *Método para cargar en memoria la cancha de "Squash".*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `void Escena::quitarCanchaSquash()`

Descripción: *Método para quitar la cancha de "Squash" del árbol y de memoria.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre `void Escena::cargarRaquetas()`

Descripción: *Método para cargar en memoria la raqueta.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void Escena::quitarRaquetas ()`

Descripción: *Método para quitar la raqueta del árbol y de memoria.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre `void Escena::cargarBola ()`

Descripción: *Método para cargar en memoria la bola.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void Escena::quitarBola ()`

Descripción: *Método para quitar la bola del árbol y de memoria.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Módulo GameCore

En este módulo es donde se encuentra el núcleo del juego y es aquí donde se toma en cuenta todos los datos necesarios para iniciar el juego y los que transcurren en el momento del juego

Nombre: `GameCore::GameCore(void)`

Descripción: *Método public de esta clase que el cual es el constructor de esta clase inicializa las variables tales como gravedad, escena, partida, frame por segundo,etc.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `class GeneralKEH : public osgGA::GUIEventHandler`

Descripción: *Clase dentro del módulo que sirve para manejar todos los eventos del teclado.*

Tipo dato retorno: *no retorna nada*.

Tipo dato entrada: *no recibe nada.*

Nombre: `GeneralKEH( GameCore* gc )`

Descripción: *Método público inicializa un* `GameCore`.

Tipo dato retorno*: no retorna nada.*

Tipo dato entrada: *no recibe ninguno.*

Nombre: `virtual bool handle`

Descripción: *Método público que sirve manejar todos los eventos que son generados por el keyboard en el juego.*

Tipo dato retorno: *no retorna nada.*

Nombre `void GameCore::Run(t_screenMode screenMode)`

Descripción: *Método público donde se inicializan todas las configuraciones y contiene las reglas del juego, aquí se encuentra el juego en si.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada*: recibe un* `t_screenMode para el seteo de la pantalla`.

Nombre: `void GameCore::frame_limiter(int fps)`

Descripción: *Método que sirve como Delimitador de FPS, así el programa correrá a un máximo de FPS sin importar que tan potente sea el sistema.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *recibe un entero con la cantidad de frames por segundo.*

Nombre: `void GameCore::calcularFPS()`
Descripción: *Método que sirve calcular los frames por segundo.*
Tipo dato retorno: *no retorna nada.*
Tipo dato entrada: *no recibe nada.*

Nombre: `bool GameCore::esperarEnFrames(int frames)`
Descripción: *Retorna falso mientras no hayan pasado los frames indicados, desde que se la llama la primera vez, una vez pasados los frames, retorna verdadero, se resetea y puede ser usada de nuevo.*
Tipo dato retorno *no retorna nada.*
Tipo dato entrada: *recibe un* `int` *del número de frames.*

Nombre `bool GameCore::esperarEnSegundos(double segundos)`
Descripción*: Método que retorna falso mientras no hayan pasado los segundos indicados, desde que se la llama la primera vez, una vez pasado el tiempo, retorna verdadero, se resetea y puede ser usada de nuevo.*
Tipo dato retorno: *no retorna un booleano.*
Tipo dato entrada: *recibe los segundos.*

Nombre: `bool GameCore::tr_Initialize ()`
Descripción: *Método que sirve para inicializar los trackers.*
Tipo dato retorno: *retorna un booleano de acuerdo a si se pudo o no inicializar.*
Tipo dato entrada: *no recibe nada.*

Nombre: `bool GameCore::tr_Connect ()`

Descripción: *Método que sirve para saber si la conección con el tracker esta lista.*

Tipo dato retorno *retorna un booleano según se haya dado la conección.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void GameCore::tr_Disconnect ()`

Descripción: *Método que sirve para desconectar el tracker.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `bool GameCore::tr_SetupDevice ()`

Descripción: *Método que sirve seteo de los sensores.*

Tipo dato retorno *retorna un booleano.*

Tipo dato entrada: *no recibe nada.*

Nombre: `bool GameCore::tr_StartCont ()`

Descripción: *Método que comienza la captura continua de los datos del sensor.*

Tipo dato retorno *retorna true si se pudo iniciar.*

Tipo dato entrada: *no recibe nada.*

Nombre: `bool GameCore::tr_StopCont()`

Descripción: *Método que sirve para detener la captura continua de los datos de los sensores.*

Tipo dato retorno *retorna true si se detuvo.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void GameCore::tr_DisplayFrame( PBYTE pBuf, DWORD dwSize )`

Descripción: *Método que sirve para obtener los valores de los trackers.*

Tipo dato retorno *no retorna nada.*

Nombre: `void GameCore::tr_DisplaySingle()`

Descripción: *Método que lee y muestra una sola muestra de los trackers.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void GameCore::tr_DisplayCont()`

Descripción: *Cada vez que se llama a esta funcion, se muestrea el estado actual de los sensores, seteados anteriormente en CONTINUO.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `osg::Vec3 GameCore::tr_getPosition (int sensor)`

Descripción*: Devuelve la posición dada por los tracker en el momento, como Vec3 de OSG.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *recibe un entero que representa un sensor.*

Nombre: `osg::Quat GameCore::tr_getAttitude (int sensor)`

Descripción*: Devuelve la orientación del sensor dado por el parametro 'sensor', como un Quat de OSG.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *recibe un entero que representa un sensor.*

Módulo jugador

Nombre: `Jugador::Jugador(int id)`

Descripción: *Método que sirve para inicializar un jugador.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *recibe un entero que es el id del jugador.*

Nombre: `Jugador::Jugador(int id, std::string nombre)`

Descripción: *Método que sirve para inicializar un jugador.*

Tipo dato retorno *no retorna nada.*

Tipo dato entrada: *recibe un entero que es el id del jugador y el nombre del jugador.*

Módulo mainApp

Nombre: `int main(int argc, char* argv[])`

Descripción: *Método donde se setea los valores de configuración inicial dela visión estereo y comienza la ejecución del proyecto.*

Módulo partida

Nombre: `Partida::Partida(int numJugadores)`

Descripción: *Método que sirve para inicializar una partida.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe el número de jugadores.*

Nombre: `Jugador* Partida::getJugador(int index)`

Descripción: *Método que sirve para obtener un jugador del arreglo de jugadores.*

Tipo dato retorno: *retorna un jugador.*

Tipo dato entrada: *recibe un* `int` *que es el índice del jugador.*

Nombre: `Jugador* Partida::getJugadorActivo()`

Descripción: *Método que sirve para obtener un jugador activo.*

Tipo dato retorno: *retorna un jugador.*

Tipo dato entrada: *no recibe nada.*

Nombre: `int Partida::getNumJugadores()`

Descripción: *Método que sirve obtener el número de jugadores.*

Tipo dato retorno: *retorna un* `int` *con el número de jugadores.*

Tipo dato entrada: *no recibe nada.*

Nombre: `int Partida::getIndexJugadorActivo()`

Descripción: *Método que sirve para obtener el índice del jugador activo.*

Tipo dato retorno: *retorna un* `int` *con el índice del jugador.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void Partida::setIndexJugadorActivo(int index)`

Descripción: *Método que sirve para setear el índice del jugador activo.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe el índice a setear.*

Nombre: `int Partida::getEstado()`

Descripción: *Método que sirve para obtener el estado de la partida.*

Tipo dato retorno: *retorna un* `int` *que es el estado de la partida.*

Tipo dato entrada: *no recibe nada.*

Nombre: `void Partida::setEstado(estadoPartida estado)`

Descripción: *Método que sirve para setear el estado de la partida.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe* `estadoPartida` *con el estado de la partida.*

Nombre: `void Partida::iniciarServicio()`

Descripción: *Método que sirve para setear el estado del jugador activo y de la partida en iniciar servicio.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Módulo plano

Nombre: `Plano::Plano( osg::Vec3 vec1, osg::Vec3 vec2, osg::Vec3 vert)`

Descripción: *Método que sirve para definir un plano con dos vectores y un vértice dado.*

Tipo dato retorno: *no retorna nada pero crea un plano.*

Tipo dato entrada: *recibe tres* `osg::Vec3 que definen los dos vectores y el vertice`.

Nombre: `Plano::~Plano()`

Descripción: *Método que sirve para remover un plano del árbol.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *no recibe nada.*

Nombre: `bool Plano::setTextura( std::string file )`

Descripción: *Método que sirve para setear la textura de un plano.*

Tipo dato retorno: *retorna un booleano que indica si se pudo poner la textura.*

Tipo dato entrada: *recibe un* `std::string` *con el nombre del archivo de la textura.*

Nombre: `bool Plano::setTexturaTransparente( std::string file, float alpha )`

Descripción: *Método que sirve para setear una textura con cierto grado de transparencia.*

Tipo dato retorno: *retorna un booleano que nos indica si se pudo poner la textura.*

Tipo dato entrada: *recibe un* `std::string` *con el nombre del archivo de la textura y un* `float` *que nos indica el grado de transparencia.*

Módulo Raqueta

Nombre: `Raqueta::Raqueta( std::string filePath )`

Descripción: *Método público que sirve para importar un archivo de raqueta dentro de la escena, además de setear el pivote y las normales de la misma.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe un* `std::string` *con el nombre del archivo del modelo.*

Nombre: `bool Raqueta::esValido()`

Descripción: *Método público que sirve para ver si los objetos raqueta son diferentes de null.*

Tipo dato retorno: *retorna el valor booleano de acuerdo existan los objetos raqueta.*

Tipo dato entrada*: no recibe ninguno.*

Nombre: `Raqueta::~Raqueta()`

Descripción: *Método público que sirve para quitar la raqueta del árbol.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada: *no recibe ningún valor*.

Nombre: `void Raqueta::agregarALaEscena(osg::Group* grupo)`

Descripción: *Método público que sirve para agregar a la escena la raqueta.*

Tipo dato retorno: *no retorna nada.*

Tipo dato entrada: *recibe un* `osg::Group al que se va adherir.`

Nombre: `void Raqueta::setPosition (const osg::Vec3 &newPos)`

Descripción: *Método público que sirve para setear la nueva posición de la raqueta.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada*: recibe un* `osg::Vec3` *con la nueva posición.*

Nombre: `void Raqueta::setOrientation (float rad1, osg::Vec3 &eje1, float rad2, osg::Vec3 &eje2, float rad3, osg::Vec3 &eje3)`

Descripción: *Método público que sirve para setear la orientación de la raqueta dependiendo de tres vectores con sus respectivos ángulos.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada: *recibe tres* `float` *que son los ángulos de cada vector y tres* `osg::Vec3` *que son los vectores respectivamente.*

Nombre: `void Raqueta::setOrientation (osg::Quat &newOrientQuat)`

Descripción: *Método público que sirve setear la nueva orientación de la raqueta.*

Tipo dato retorno: *no retorna ningún valor pero cambia la orientación de la raqueta.*

Tipo dato entrada: *recibe un* `osg::Quat` *que es un quartation con la nueva posición de la raqueta.*

Nombre: `void Raqueta::setCallback( osg::NodeCallback *nc )`

Descripción: *Método público que sirve para* setear *el nodo* callback *actualizado.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada: *recibe un puntero a un* `osg::NodeCallback`.

Nombre: `void Raqueta::setVolumen( float volumen )`

Descripción: *Método que sirve para setear el volumen de la raqueta y con esto encontrar el valor de la masa de la misma.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada: *recibe un* `float` *con el valor del volumen*.

Nombre: `void Raqueta::setDensidad( float dens )`

Descripción: *Método que sirve para setear la densidad de la raqueta y con esto encontrar el valor de la masa de la misma.*

Tipo dato retorno: *no retorna ningún valor.*

Tipo dato entrada: *recibe un* `float` *con el valor de la densidad*.

Módulo Sonido

Nombre: `bool Sound_Init()`

Descripción: *Método que sirve para inicializar la librería SDL para el audio.*

Tipo dato retorno: *retorna un booleano que indica si se pudo o no inicializar.*

Tipo dato entrada: *no recibe nada.*

Módulo Texto

Nombre: `TextBox::TextBox(unsigned int id)`

Descripción: *Método que sirve para posicionar un mensaje en la pantalla.*

Tipo dato retorno: *no retorna ningún dato.*

Tipo dato entrada: *recibe un índice*

Nombre: `void TextBox::setText(const string& t)`

Descripción: *Método que sirve para setear el texto dentro del* `textbox`.

Tipo dato retorno: *no retorna ningún dato.*

Tipo dato entrada: *recibe un* `string` *que es el texto a introducir.*

Nombre: `void TextBox::setTextSize(unsigned int size)`

Descripción: *Método que sirve para setear el tamaño del texto.*

Tipo dato retorno: *no retorna ningún dato.*

Tipo dato entrada: *recibe un* `int` *con el tamaño del texto.*

Nombre: `osg::Vec3 TextBox::getTextObjectSize()`

Descripción: *Método que retorna el tamaño del objeto texto dibujado como un vector donde las coordenadas significan la dimensión en x,y,z.*

Tipo dato retorno: *retorna un* `osg::Vec3` .

Tipo dato entrada: *no recibe ningún dato.*

Nombre: `osg::Vec2d TextBox::getPosition()`

Descripción: *Método que retorna el tamaño del objeto texto dibujado como un vector donde las coordenadas significan la dimensión en x,y solo son dos coordenadas ya que es cuando utilizamos el cuadro de texto en un Hud.*

Tipo dato retorno: *retorna un* `osg::Vec2d` .

Tipo dato entrada: *no recibe ningún dato.*

REFERENCIAS

[1] Hickok, Ralph. "Squash Rackets Rules". Hickok Sports, http://www.hickoksports.com/rules/rsquashr.shtml. Fecha de consulta: 25 de febrero de 2010.

[2] Bellamy, Rex. "The Story of Squash". Cassell Ltd, Londres. ISBN 0-304-29766-6. Año de publicación: 1978.

[3] Davis, Erik. "Techgnosis: Myth, magic and mysticism in the information age". Five Star. ISBN 1852427728. Año de publicación: 2005.

[4] Berchek, Chad. "2-Dimensional Elastic Collisions without Trigonometry". http://www.vobarian.com/collisions/. Fecha de consulta: 17 de noviembre de 2009.