

Dunas de Arrakis

Etapa Final

Danilo Abellá, Sergio Salinas

Resumen—Se presenta el primer prototipo del desarrollo de un videojuego basado en la novela de ciencia ficción Dune de Frank Herbert, este videojuego esta siendo programado en el motor gráfico Unity con el lenguaje de programación Javascript con Blender para la creación de los modelos 3D.

Index Terms—Videojuego, Computación Gráfica, Dune, Unity.

1. INTRODUCTION

Los avances en la computación gráfica han permitido que cualquiera pueda programar su propio videojuego sin mayores conocimientos en programación más que lo básico, además gracias a los nuevos software de modelado se puede tener un detalle casi realista cuando se debe representar el mundo real, e incluso permite crear mundos ficticios de que los que sea nunca poder ver más allá de la imaginación. Debido a esto este informe tratará sobre la creación de un videojuego basado en la novela de ciencia ficción Dune de Frank Herbert.

El inicio de todo videojuego son los modelos a utilizar en él, una vez creados se puede empezar con los movimientos y la interacción de los modelos. Este informe trata acerca de los modelos principales del videojuego en desarrollo Dunas de Arrakis, los modelos son el tanque y la bala, además se muestra el script movimiento de ambos.

2. ACERCA DE DUNE

Dune es una novela de ciencia ficción escrita por Frank Herbert en 1965 [2], la historia se desarrolla 20000 años en el futuro, cuenta la historia del joven del joven Paul Atreides, heredero del ducado de la Casa Atreides, luego de que su padre el Duque Leto recibe la orden del emperador para transladarse al planeta Arrakis y encargarse de la extracción de la especia.

La especia es una sustancia única el universo que se extrae solo del planeta Arrakis y la más importante debida a que tiene la propiedad de extender la vida de quién lo que consume, además de otorgar el poder la presciencia, que es fundamental para los viajes intergalácticos.

Debido a la alta ganancia que genera la especia la casa Atreides se ve constantemente enfrentada a las otras grandes casas que buscan hacerse con el poder de los derechos de extracción de la especia para aumentar su economía. Los principales enemigos de los Atreides con los que estos se ven enfrentados constantemente es la casa Harkonnen.

3. OBJETIVOS

3.1. Objetivo general

Desarrollar e implementar un programa en escrito en JavaScript usando el motor de videojuegos Unity y el software Blender para la creación de un videojuego basado en la novela Dune de Frank Herbert.

3.2. Objetivos específicos

- Diseñar los bocetos de los modelos
- Utilizar los temas vistos en clases
 - Translación
 - Rotación
 - Creación y destrucción en tiempo real
 - Escalamiento

4. DESCRIPCIÓN DEL PROBLEMA

4.1. Motivación

El universo que Frank Herbert creo con su saga de libros Dune ha sido de gran inspiración para los integrantes del grupo, aprovechar las capacidades de la computación gráfica para poder representar este mundo propone un desafío interesante para estos.

4.2. Definición

El videojuego consiste en que el usuario controla un tanque y desde un punto inicial debe llegar hasta un punto final para ganar. Para llegar a dicha meta debe atravesar obstáculos (tanques enemigos, piedras, etc). Este tanque le pertenece a la casa Atreides y en el trayecto es atacado por tanques de guerra pertenecientes a la casa Harkonnen. El escenario son las dunas del planeta Arrakis, este escenario presenta rocas y las mismas dunas para ayudar al tanque a defenderse de los ataques de sus enemigos. Los tanques a sus ves pueden disparar balas que al colisionar con otros tanques le restan 1 de vida (vida inicial: 3). Cuando la vida llega a 0 el objeto tanque se destrulle.

4.3. Estado del arte

Existen varios videojuegos basados en tanques y en dune, los siguientes dos son los más importantes ya que es el se basarán los modelos a utilizar en el juego y el segundo el movimiento del tanque en terreno desnivelado.

- Dune 2000, es un juego de estrategia para PC basado en el universo de dune, este videojuego muestra una gran cantidad de arte conceptual en el que se va a basar el videojuego para los modelos 3D.
- Tank Attack 3D, juego en flash de navegador que simula el viaje de un tanque a través de un terreno desnivelado. Se basara en él para simular el movimiento del tanque.

5. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

5.1. Características de la solución

La solución contempla usar el software Blender para crear los modelos 3D y luego implementarlos en el software de motor gráfico Unity 3D, dentro de este programar en lenguaje Javascript las interacciones de los objetos con el usuario y entre los modelos mismos.

6. PROPÓSITO DE LA SOLUCIÓN

El propósito es la implementación y aprendizaje de las materias vistas en la clase de Computación gráfica.

7. ALCANCES Y LIMITACIONES DE LA SOLUCIÓN

- Solo el tanque que controla el usuario tiene la propiedad de traslación
- No es de mundo abierto, el usuario solo viajar por los lugares que el juego propone.
- El videojuego es de un solo jugador.

8. HERRAMIENTAS Y AMBIENTE DE DESARROLLO

8.1. Herramientas de desarrollo

Para el desarrollo del software se usaran dos notebooks, uno por cada integrantes, en la tabla 1 se muestran las características de cada uno.

Tabla 1
Herramientas de Hardware

AMD® Turion™ X2 Dual-Core Mobile RM-72 2.10GHz	4GB
Intel Core i7-6500 2.59GHz	8GB

El software a utilizar se muestra en la siguiente la tabla

2

Tabla 2
Herramientas de software

Motor de gráfico	Unity
IDLE	Unity
Modelado	Blender
Lenguaje de programación	Javascript
Sistema Operativo	Windows 8

8.2. Ambiente de desarrollo

El videojuego será desarrollado en su totalidad en la biblioteca central de la Universidad Santiago de Chile.

8.3. Recursos humanos

Los recursos humanos son los dos integrantes del grupo, que se encargarán en conjunto de la documentación y el desarrollo del videojuego.

9. METODOLOGÍA

En vista del poco tiempo disponible, que el videojuego a crear es simple y el equipo de trabajo son solo dos integrantes, se decidió utilizar el modelo Lineal secuencial, este consiste en 4 etapas principales [1], Análisis de requisitos, Diseño, implementación y pruebas, los tiempos en la que se lleva cada etapa se muestra en la carta Gantt en el Apéndice A.

9.1. Modelado tanque

La realización del tanque fue realizada en Blender y consta de 3 partes:

1. Boceto del tanque (fig:1)

Lo primero es un boceto del diseño del tanque en 2D desde distintas perspectivas, para poder tener una mejor noción de tamaño, forma y dimensión mas exacta, y así tener una base mucho mas sólida para que la etapa de modelado sea mucho mas fácil.

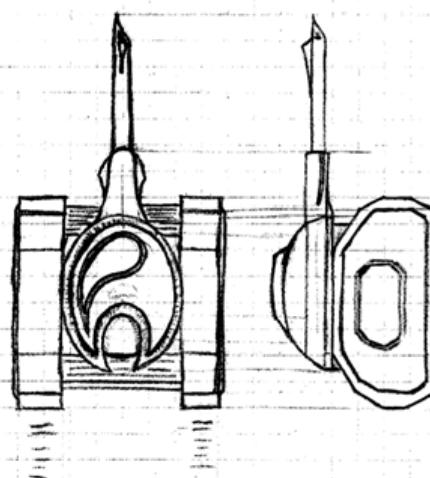


Figura 1. Boceto tanque

2. Modelado del tanque (fig:5,fig:4)

Una vez echo el boceto se procede a modelar

el tanque en Blender desde cada una de sus perspectivas en el eje X,Y e Z.

3. Texturizado del tanque (fig:2,fig:6,fig:7)

Finalmente se procede a utilizar un archivo de imagen (fig:2) para asignar una textura o más al modelo del tanque. La textura del tanque se muestra en la figura 2.

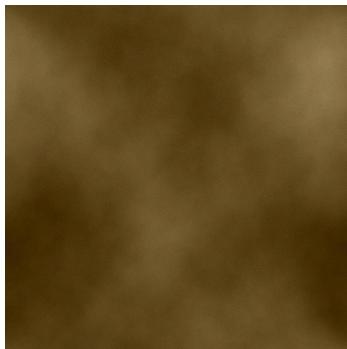


Figura 2. Textura del tanque

9.2. Movimiento tanque

Para el movimiento del tanque se aplicaron tanto las transformaciones de traslación como rotación (Apéndice A).

Para simular el desplazamiento del tanque se implementó la traslación sólo sobre el eje Z, tanto positiva como negativamente.

En cuanto a la rotación, se le aplicó rotación en torno al eje Y para simular el virar del tanque el cual sólo será efectuada cuando el tanque esté en movimiento, para poder hacer mas realista el desplazamiento del tanque.

9.3. Bala

La realización de la bala consta de las siguientes partes:

1. Modelado de la bala (fig:8)

La bala es representada como una primitiva de esfera, para la cual se le aplicó un texturizado con una imagen .png.

2. Interacción con el entorno (Apéndices B, C, D, E)

Los pasos que la bala pasa dentro del juego son, su creación, la cual se instancia desde el cañón del tanque y sólo cuando el usuario acciona una tecla correspondiente o salga cada cierto tiempo desde un tanque enemigo.

Una vez creada la bala esta automáticamente se mueve en el eje z hasta colisionar con un tanque o (en la mayoría de los casos) eliminarse tras pasar 6 segundos dentro del juego.

En caso de colisionar con un tanque, si éste era el jugador el juego se termina y el tanque se elimina, en caso de ser un tanque enemigo sólo se elimina el tanque enemigo.

9.4. Terreno

El terreno es representado como un heightmap, o sea, un mapa de alturas o un campo de alturas la cual es una imagen ráster que se usa para almacenar valores, como los datos de elevación de superficies, para mostrarlos en gráficos 3D.

Utilizando Photoshop se creó el heightmap (fig:3) para el terreno en forma de dunas.

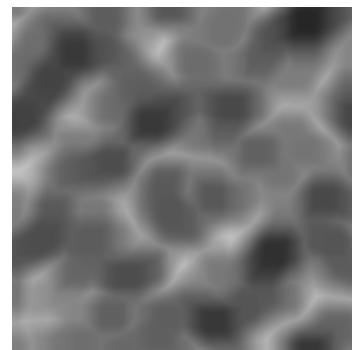


Figura 3. heightmap de dunas

Luego fue solo importar el heightmap a Unity para poder así crear el terreno 3D.

10. RESULTADOS

Las etapas del modelo del tanque se pueden ver en la figura 4 y 5.

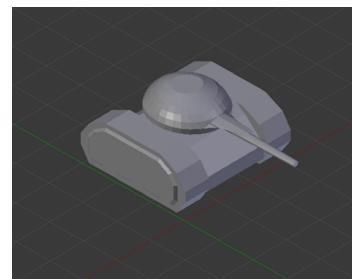


Figura 4. Modelo tanque sin texturas

El modelo del tanque con texturas se puede apreciar en las figuras 6 y 7

El modelo de la bala con texturas es la figura 8.

La figura (9) representa el terreno de las dunas.

11. USOS POTENCIALES

Para jugadores casuales o fanáticos, pero más que nada para estos últimos, donde pasar el rato jugando un juego que represente una escena de sus libros favoritos, es de seguro, algo que a más de un curioso le va a interesar.

También para algún jugador casual que por mera curiosidad pruebe el juego y conozca o no la temática de Dune.

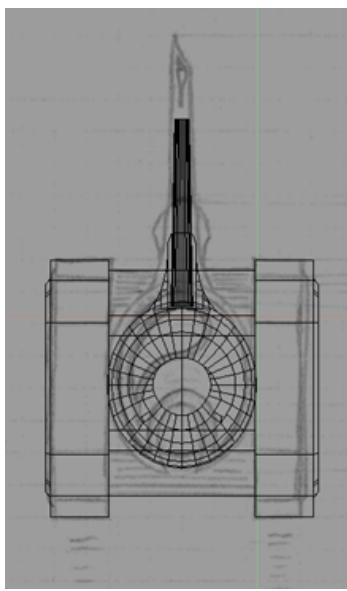


Figura 5. Modelo tanque sin texturas visto desde arriba

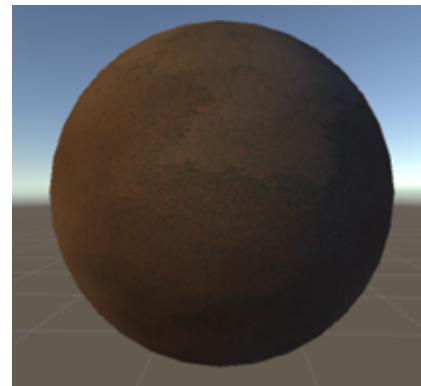


Figura 8. Bala con texturas

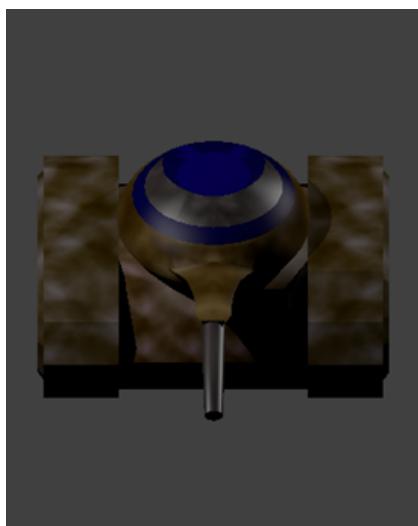


Figura 6. Modelo tanque con texturas

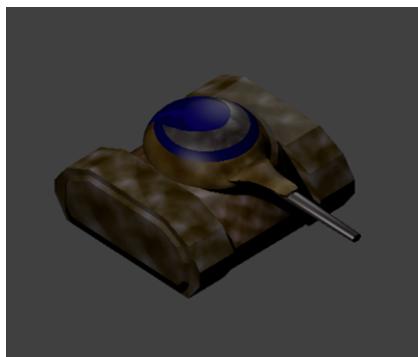


Figura 7. Modelo tanque con texturas

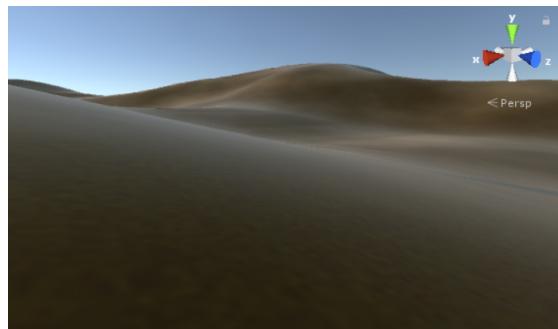


Figura 9. Terreno

12. MANUAL DE USUARIO

Una vez ejecutado el programa desde su archivo exe, se mostrará en pantalla un menú con 2 pestañas (fig:10), "Graphics" para configuraciones gráficas del juego y comenzar a jugar, y "Input" para modificar los controles del juego.

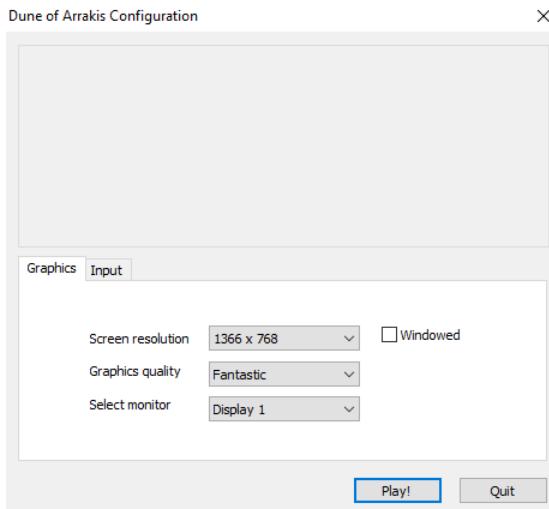


Figura 10. Configuración

Las teclas para jugar por defecto son:

- ↑: Avanzar.
- ↓: Retroceder.
- ←: Virar hacia la derecha.
- : Virar hacia la derecha.
- A: disparar.

Una vez dentro del juego aparecerá un menú con 2 botones (fig:11), "Empezar" para comenzar a jugar y "Salir" para cerrar el juego.



Figura 11. Menú

Una vez in-game (fig:12), el jugador tiene que llegar hasta la luz azul para poder ganar, para lograrlo tiene que recorrer el mapa esquivando las balas de tanques enemigos y matándolos.



Figura 12. Menú

13. CONCLUSIÓN

El uso de un motor gráfico para la creación de un videojuego es de gran ayuda cuando se requiere simular ambientes complejos con gran realismo, en base a esto se pudo analizar de mejor forma como planificar el proyecto y la idea que se desea simular. Con esto se da por finalizado esta parte del anteproyecto y se prosigue con el diseño y la codificación.

El modelado y el movimiento es la base principal de un videojuego, debido a esto se decidió que esto sería lo primero que se implementaría en un prototipo. Gracias a la herramientas de desarrollo actual como Unity y Blender se pudo agilizar ambas tareas, gracias a esto se ha dejado lista la base del videojuego y completamente preparado para la entregar el videojuego completo en su fecha.

REFERENCIAS

- [1] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, revised. Palgrave Macmillan, 2005.
- [2] Frank Herbert, *Dune, Volume 1 of Dune Series*, revised. Ace Books, 2005.

APÉNDICE A**CÓDIGO DE MOVIMIENTO DEL TANQUE (TRASLACIÓN Y ROTACIÓN)**

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Movimiento : MonoBehaviour {

    public float moveSpeed = 10f;
    public float turnSpeed = 50f;

    void Start () {}

    void Update () {

        // Traslaciones
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(new Vector3( 0, 0, -moveSpeed) * Time.deltaTime);

            // Rotaciones
            if (Input.GetKey(KeyCode.LeftArrow))
                transform.Rotate(Vector3.up, -turnSpeed * Time.deltaTime);

            if (Input.GetKey(KeyCode.RightArrow))
                transform.Rotate(Vector3.up, turnSpeed * Time.deltaTime);
        }

        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(new Vector3(0, 0, moveSpeed) * Time.deltaTime);

            // Rotaciones
            if (Input.GetKey(KeyCode.LeftArrow))
                transform.Rotate(Vector3.up, turnSpeed * Time.deltaTime);

            if (Input.GetKey(KeyCode.RightArrow))
                transform.Rotate(Vector3.up, -turnSpeed * Time.deltaTime);
        }
    }
}

```

APÉNDICE B**CÓDIGO DE CREACIÓN DE LA BALA Y ASIGNACIÓN DE TRASLACIÓN EN EL EJE Z.**

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Disparo : MonoBehaviour
{
    public Rigidbody rocketPrefab;
    public Transform barrelEnd;

    void Update()
    {
        if (Input.GetKeyDown("a"))
        {
            Rigidbody rocketInstance;
            rocketInstance = Instantiate(rocketPrefab, barrelEnd.position, barrelEnd.rotation)
        }
    }
}

```

```

        as Rigidbody;      // Función de instanciar bala
        rocketInstance.AddForce(barrelEnd.forward * 5000);
    }
}
}
}
}
```

APÉNDICE C**CÓDIGO DE INTERACCIÓN DE LA BALA CON EL TANQUE ENEMIGO.**

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Bala : MonoBehaviour {

    void OnCollisionEnter(Collision col) {

        if (col.gameObject.tag == "Enemigo")
        {
            Destroy(col.gameObject);
        }
    }
}
```

APÉNDICE D**CÓDIGO DE INTERACCIÓN DE LA BALA ENEMIGA CON EL JUGADOR.**

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BalaEnemiga : MonoBehaviour {

    public float cont;

    private void Start()
    {
        cont = 0;
    }

    void OnCollisionEnter(Collision col)
    {

        if (col.gameObject.tag == "Player")
        {
            Destroy(col.gameObject);
        }
    }

    private void Update()
    {
        cont = cont + 1 * Time.deltaTime;
        if (cont > 6 )
        {
            Destroy(this.gameObject);
        }
    }
}
```

APÉNDICE E**CÓDIGO DISPARO DE TANQUES ENEMIGOS.**

Además de esta hay otra versión donde las variables **cont** y **frec** son 1 y 2 respectivamente.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DisparoEnemigo : MonoBehaviour {

    public Rigidbody rocketPrefab;
    public Transform barrelEnd;

    public float cont, frec;

    private void Start()
    {
        cont = 0;
        frec = 3;
    }

    void Update()
    {
        cont = cont + 1 * Time.deltaTime;

        if (cont > frec)
        {
            cont = 0;
            Rigidbody rocketInstance;
            rocketInstance = Instantiate(rocketPrefab, barrelEnd.position, barrelEnd.rotation) as Rigidbody;
            rocketInstance.AddForce(barrelEnd.forward * 5000);

        }
    }
}
```

APÉNDICE F**CÓDIGO DISPARO DE TANQUES ENEMIGOS.**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CameraView : MonoBehaviour {

    [SerializeField]
    Camera FirstPCamera;
    [SerializeField]
    Camera SecondPCamera;

    public Text Muerto;
    public int newFontSize;
    public string Perder;
    public string Ganar;

    public Color newColor;

    // Use this for initialization
    void Start ()
    {
```

```
FirstPCamera.GetComponent<Camera>().enabled = true;
SecondPCamera.GetComponent<Camera>().enabled = false;
}

void OnCollisionEnter(Collision col)
{

    if (col.gameObject.tag == "Bala_Enemiga")
    {
        FirstPCamera.GetComponent<Camera>().enabled = false;
        SecondPCamera.GetComponent<Camera>().enabled = true;
        Muerto.text = Perder;
        Muerto.fontSize = newFontSize;
        Muerto.color = newColor;
    }

    if (col.gameObject.tag == "Finish")
    {
        FirstPCamera.GetComponent<Camera>().enabled = false;
        SecondPCamera.GetComponent<Camera>().enabled = true;
        Muerto.text = Ganar;
        Muerto.fontSize = newFontSize;
        Muerto.color = newColor;
    }
}
```

APÉNDICE G

CARTA GANTT

