|  |  |  |
| --- | --- | --- |
| Texto  Descripción generada automáticamente con confianza media | TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERÍA  Universidad Nacional de Jujuy |  |

Trabajo Práctico

N°3

Tactaca, Cecilia Yazmin – LU TUV000764

*Profesores:*

*Mg. Ing. Ariel Alejandro Vega*

*Año 2024*

**FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS**

Índice

[Uso de listas, colisiones y deltaTime 3](#_Toc169290120)

[Punto 1: 3](#_Toc169290121)

[Desarrollo del ejercicio 1: 3](#_Toc169290122)

[Fase de análisis: 3](#_Toc169290123)

[Fase de diseño: 3](#_Toc169290124)

[Uso de componentes e imágenes 3](#_Toc169290125)

[Punto 2: 3](#_Toc169290126)

[Desarrollo del ejercicio 1: 4](#_Toc169290127)

[Fase de análisis: 4](#_Toc169290128)

[Fase de diseño: 4](#_Toc169290129)

[Fuentes bibliográficas 4](#_Toc169290130)

# Uso de listas, colisiones y deltaTime

Punto 1:

Realice un diagrama de clases y programe un juego con la clase Snake. Snake está compuesto por una cabeza y un cuerpo. El cuerpo es un conjunto de “cuadrados”. Así, Snake “está compuesto por 1 o muchas partes del cuerpo” donde cada parte es un cuadrado. En el escenario aparece de manera aleatoria un animal. Existen 3 tipos de animales: ratones, pajaritos e insectos (Ud. elige cual será el ejemplo que use para cada uno de estos tipos). Cuando la cabeza del Snake se come a este animal, aparece otro en otra posición aleatoria y además crece el cuerpo del Snake (se agrega otro cuerpo). Cada animal tiene un puntaje que adquiere el Snake cuando se lo come. Los animales deben ser dibujados usando imágenes. Snake puede dibujarse usando imágenes o primitivas. Se debe visualizar el puntaje en todo momento. El juego se termina cuando ha pasado un minuto. Al comerse el animal se debe guardar en una lista, el animal que se ha comido, y el tiempo en el que se lo comió y mostrarlo en pantalla. La velocidad con la que se mueve Snake debe normalizarse mediante el uso de DeltaTime.

## Desarrollo del ejercicio 1:

### Fase de análisis:

**TARJETA DE HISTORIA DE USUARIO**

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Código:** HU006 | **Usuario:** Jugador |
| **Nombre de historia de usuario:** Construcción de escenario y ubicación de GameObject. | |
| **Prioridad:** Alta | **Riesgo de desarrollo:** Alta |
| **Estimación:** 1 hora | **Iteración asignada:** 1 |
| **Responsable:** Tactaca, Cecilia Y. | |
| **Descripción:** | |
| **Criterios de aceptación:** | |
| **Observaciones:** | |

### Fase de diseño:

**ESTRUCTURA DE APLICACIÓN**

# Uso de componentes e imágenes

Punto 2:

Diagrame y programe un conjunto de muros ubicados en diferentes lugares de la pantalla. Cada muro tiene un componente que representa una imagen. Además, el muro tiene una resistencia indicada por un valor que va de 10 a 30 puntos. En la parte inferior tenemos un Tanque, también tiene un componente para su imagen. Tanto tanque como Muro tienen un componente Transform para la ubicación. El conjunto de muros se almacena en una clase GestorMurallas, que está formado (agregación) por una lista (arrayList) de muros. Tanque dispara balas, estas balas se almacenan en una lista de balas. El gestorMurallas tiene un método verificarColision() que recibe como parámetro una lista de balas, y se encarga de verificar si alguna de las balas de esa lista ha colisionado con el muro. Cada bala que colisiona con un muro, lo debilita en 10 puntos y cuando ya no tiene resistencia, lo destruye. Aplique el deltaTime para la velocidad de balas y la velocidad del tanque (que solo se puede mover hacia los costados).

## Desarrollo del ejercicio 1:

### Fase de análisis:

**TARJETA DE HISTORIA DE USUARIO**

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Código:** HU001 | **Usuario:** Jugador |
| **Nombre de historia de usuario:** Construcción de escenario y ubicación de GameObject. | |
| **Prioridad:** Alta | **Riesgo de desarrollo:** Alta |
| **Estimación:** 1 hora | **Iteración asignada:** 1 |
| **Responsable:** Tactaca, Cecilia Y. | |
| **Descripción:** | |
| **Criterios de aceptación:** | |
| **Observaciones:** | |

### Fase de diseño:

**ESTRUCTURA DE APLICACIÓN**

# Fuentes bibliográficas

* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Expresiones aritméticas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Bifurcaciones. Expresiones Lógicas. Estructuras selectivas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. DISEÑO DE ALGORITMOS: Estructuras Selectivas. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* Mg. Ing. Vega, A. Programando con Processing. Facultad de Ingeniería. Universidad Nacional de Jujuy.
* <https://www.youtube.com/watch?v=EAclc8NcnOo&t=1s>