

**Tecnicatura Universitaria en Diseño Integral de Videojuegos**

**FPOO**

**Trabajo Práctico N°2**

**Alumno:** Mateo Nicolás Goitea

**DNI:** 45977040

**LU:** TUV000720

**Profesor:** Ariel Alejandro Vega

**Año:** 2024

**Punto 1:** Desarrolle una historia de usuario, en la cual defina la visualización y movimiento de una clase GameObject, de la que heredan Shooter y Asteroide. GameObjects es abstracta, y posee atributos protegidos: posición, imagen; además del método abstracto display() y mover(). Además debe poseer un HUD que visualice la cantidad de vidas del Shooter. Utilce un JoyPad para generar los movimientos.

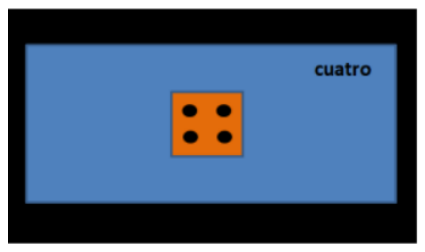
**Desarrollo:**

Como jugador quiero poder visualizar en la pantalla mi nave y los asteroides en movimiento, ambos con sus respectivos sprites. También quiero que se visualice mi cantidad de vidas a modo de interfaz y quiero poder mover la nave con el teclado en todas las direcciones. Requiero esto para poder ubicar correctamente mi nave y los asteroides en el escenario, y para controlar la nave para esquivar los asteroides, y para ver la cantidad de vidas que tengo y saber que tanto puedo arriesgarme en el juego.

|  |  |
| --- | --- |
| HISTORIA DE USUARIO | |
| **Código:** HU001 | **Usuario:** Jugador |
| **Nombre de Historia de Usuario:** Construcción de escenario y Game Objects | |
| **Prioridad:** Alta | **Riesgo de Desarrollo:** Alta |
| **Estimación:** 1 hora | **Iteración asignada:** 1 |
| **Descripción:** Como jugador quiero poder visualizar en la pantalla mi nave y los asteroides en movimiento, ambos con sus respectivos sprites. También quiero que se visualice mi cantidad de vidas a modo de interfaz y quiero poder mover la nave con el teclado en todas las direcciones. Requiero esto para poder ubicar correctamente mi nave y los asteroides en el escenario, y para controlar la nave para esquivar los asteroides, y para ver la cantidad de vidas que tengo y saber que tanto puedo arriesgarme en el juego. | |
| **Criterios de aceptación:**  El jugador debe ser una nave y debe poder moverse en todas las direcciones con el teclado.    Asteroides aparecen en la parte superior de la pantalla aleatoriamente descendiendo a modo de obstáculos para el jugador.    En la pantalla deben aparecer la cantidad de vidas del jugador, las cuales se reducirán al recibir el impacto de un asteroide. | |
| **Observaciones:** En este modelo no se considera la rigidez de los límites de la pantalla. | |

**Punto 2:** Desarrolle un videojuego que cumpla con las siguientes especificaciones:

Realice un diagrama de clases



Como se observa se trata de un dado. El cual al presionar un botón debe generar un número aleatorio entre 1 y 6 y dibujarlo. Además, debe mostrar el número en la parte superior derecha. Repetir esto cuantas veces lo desee y al finalizar (con otro botón) debe dibujar por consola y agrupado en filas de 4 columnas los dados obtenidos.

Al momento de programar utilice constructores sobrecargados. Considere que el dado se muestra en un tablero, este tablero contiene al dado, y al texto.

Además, almacene cada dado obtenido en un arreglo. Considere aplicar la herencia respecto de que existe una clase abstracta padre GameObject, de la que hereda la posición y el método abstracto display(). Luego recrear otra versión donde use imágenes en lugar de dibujar con las primitivas.

**Punto 3:** Realice el modelado de las clases que intervienen en el juego frogger a partir de la

Fig. 1. Realice la construcción de las clases en processing. El juego debe llegar a poder mostrar en pantalla la visualización de los diferentes objetos modelados. Utilice herencia y encapsulamiento para los vehículos. Además, los vehículos deben guardarse en una lista de objetos que es atributo de la clase SpawnerVehiculos.

**Punto 4**

Considere programar un juego de naves. Debe usar imágenes para las naves, los asteroides y los enemigos. Aplique herencia. Use una interface denominada IDisplayable que tenga el método display(). Defina dos interfaces más: IMoveable que tenga el método mover() y Otra IControler que tenga el método readCommand();

Usando el sentido común haga que las clases Nave, Asteroid y Enemy implementen las interfaces correspondientes. Finalmente use la dependencia para que la nave dispare balas que serán almacenadas en una lista de balas. Las balas se deben destruir cuando salen de pantalla.



Figura 1. Modelo juego Frogger