Trabajo Práctico N°2 – Raskovsky David Adrián

***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.

|  |  |
| --- | --- |
| **Historia de Usuario: TP2Punto1 (**Visualización y Movimiento en un Juego Espacial) | |
| *Identificador****: UH\_TP2-Pto.01*** | *Usuario****: Usuario*** |
| *Prioridad en negocio****: -*** | *Riesgo en desarrollo****: -*** |
| *Puntos estimados****: -*** | *Iteración asignada: -* |
| *Programador Responsable****: Raskovsky David Adrián*** | |
| **DESCRIPCIÓN (como,quiero,para que):**  **Como** usuario de aplicaciones de videojuegos, **quiero** controlar la nave llamada Shooter, con un JoyPad mover su posición en la pantalla **para** esquivar el Asteroide, y ver el estado en el Hud la cantidad de vidas restantes. | |
| **OBSERVACIONES:**   * El juego inicia y aparace Shooter en el lienzo en una posición dada por el JoyPad conducido por el jugador. * Aparece un Asteroide que se mueve de arriba hacia abajo y cruza horizontalmente la pantalla entrando por cualquier punto y sale por abajo de la pantalla y si choca con la nave Shooter esta muere o desaparece. * El HUD informa al jugador sobre las vidas restantes y solo hay tres objetos activos en la pantalla. * Se debe decidir si se usara la clase PVector para manejo de la velocidad o la clase PImage para imágenes o no. | |
| **DATOS DE ENTRADA:** | |
| * Tamaño de lienzo (size (800,600)) * Variable cantidad total de vidas de tipo entera: (int totalVidas) //es la cantidad inicial y total de vidas . * Variable contador de vidas de tipo entera: ( int contadorVidas= contadorVidas – totalVidas) * Posición(x,y) dada por el Control del Joystick para el objeto nave. * Posición del HUD (x,y) en la parte superior del lienzo mostrando el estado del Juego. * Contador de vidas en cero, El HUD muestra la cantidad actual de vidas de la Shooter. * Posición (x,y) para el asteroide al nacer en la parte superior del lienzo.   **DATOS DE SALIDA:** | |
| Visualización en la Pantalla:   * Representación de la Shooter (triángulo equilátero). * Representación del asteroide (círculo). * Visualización del HUD mostrando vidas restantes = 0. * Mensaje de "Game Over" al finalizar en el centro de la pantalla. | |
| **PORCESOS:**  Inicialización de variables de entrada.  Crear una instancia del objeto nave= new Shooter.  Crear una instancia del objeto asteroide = new Asteroide.  Crear una instancia del objeto hud = new HUD.   * PROCESO Leer las entradas del joystick. * POCESO Mover la Shooter según la entrada del joystick. * PROCESO Mover asteroides de arriba hacia abajo. * PROCESO Actualizar Movimiento de Asteroide. * PROCESO Actualizar Movimiento de la Shooter.   ***Verificación de Colisiones***:   * PROCESO Coliciones   + Si la distancia entre la Shooter y el asteroides es cero(0) o menor.   + Si hay colisión   Contador de vidas menos 1 //Reducir una vida.   * PROCESO Actualizar HUD   + Si hay colisión Refrescar el HUD con mermando una vida.   ***Fin del Juego***:   * PROCESO Controlar vidas   Si contador de vidas == 0 //si el contadorVidas== 0  MOSTRAR "Game Over" en el centro de la pantalla. | |

|  |
| --- |
| **CRITERIOS DE ACEPTACIÓN** |
| * La Shooterdebe poder moverse correctamente aleatoriamente, usando el Joypad. * La Shooter debe ser fluida en la pantalla sin cortes cuando se mueve. * El HUD debe mostrar correctamente la cantidad de vidas de la Shooter en todo momento. * La Shooter debe desaparecer al chocar o tocar el Asteroide. * El contador de vidas en el Hud merma una vida al chocar o tocar el Asteroide con la Shooter o viceversa. * El Asteroide debe generarse en una posición aleatoria en la parte superior del lienzo, al iniciar el juego. * El Asteroide debe moverse verticalmente hacia abajo, continuamente y con velocidad constante sin interrup ciones y desaparecer en la parte inferior del lienzo. * El Asteroides debe seguir su curso al choca o no al Shooter, hasta desaparecer en el fondo del lienzo. * El HUD aparece en posición arriba del lienzo y debe mostrar la cantidad total de vidas del Shooter. * La visualización del HUD debe ser clara y legible, permitiendo ver rápidamente cuántas vidas le quedan. * El HUD debe actualizarse automáticamente y cada vez que el Shooter muere mostrando el número correcto de vidas restantes. * El juego termina cuando el contador de vidas llegue a cero vidas, la Shooter ya no tiene vidas (contVidas=0) se debe mostrar mensaje de juego terminado o “GAME OVER”. |
| **FASE DE DISEÑO:** |
| **Diagrama de clases estructura** |
| .-  .-- |
|  |
|  |
| **PSEUDOCODIGO:**  **Clase Ejercicio9**  Rectangulo rect; // donde rect es variable puntero a la clase Rectángulo.  **Método *setup***()  ***Inicio***  //***Variables***  anchoRect  int altoRect  int espEntRect  ***fin***  **Proceso**  *Size(440,420); //se define tamaño de lienzo*  *Leer anchoRect = 40;*  *Leer altoRect = 20;*  *Crear rect // crear el objeto rect = new rectangulo(anchoRect,altoRect,espEntRect); //con el constructor*  **Llamado a función**  *dibujarRectangulo();*  **FinProceso**  **Fin método setup()**  **Función** *dibujarRectangulo();*  *rect.dibujarPared(); // llama a método dibujarPared() en la clase rect.*  **Fin función.**  **Fin clase Ejercicio9 .**  **CLASE Rectangulo()**  **Atributos::**  private anchR, altoR, espaR  private PVector coordenadasR; |
| **Método dibujarPared()** //en la clase Rectangulo  for (desde coordenadasR.x+20; hasta width; con paso (anchR + espaR))  for (desde coordenadasR.y+20; hasta height; con paso (altoR + espaR))  rect(x,y,anchR,altoR); //Primitiva grafica el rectangulo en la posicion x,y  end for  end for  **fin método dibujarPared()**  **Fin CLASE Rectangulo()** |

**Historia de Usuario: Visualización y Movimiento de GameObject**

**Título:** Visualización y Movimiento de GameObject en un Juego Espacial

**Descripción:** Como jugador, quiero controlar un objeto en el juego (Shooter) utilizando un JoyPad para moverlo y visualizar su posición y estado en la pantalla, de manera que pueda interactuar con otros objetos (Asteroides) y ver la cantidad de vidas restantes en un HUD.

**Criterios de Aceptación:**

1. **Clase Abstracta GameObject:**
   * **Atributos Protegidos:**
     + posición: Coordenadas (x, y) del objeto en el espacio del juego.
     + imagen: Representación visual del objeto.
   * **Métodos Abstractos:**
     + display(): Método para mostrar el objeto en la pantalla.
     + mover(): Método para actualizar la posición del objeto.
2. **Clase Shooter (Hereda de GameObject):**
   * **Atributos:**
     + vidas: Número de vidas del Shooter.
   * **Métodos:**
     + display(): Implementación específica para mostrar el Shooter.
     + mover(): Implementación específica para mover el Shooter basado en la entrada del JoyPad.
3. **Clase Asteroide (Hereda de GameObject):**
   * **Métodos:**
     + display(): Implementación específica para mostrar el Asteroide.
     + mover(): Implementación específica para mover el Asteroide de manera autónoma.
4. **HUD (Heads-Up Display):**
   * Visualización de la cantidad de vidas del Shooter en la pantalla.
5. **JoyPad:**
   * Utilizado para generar los movimientos del Shooter.
   * Debe permitir mover el Shooter en las direcciones básicas (arriba, abajo, izquierda, derecha).

**Escenario:**

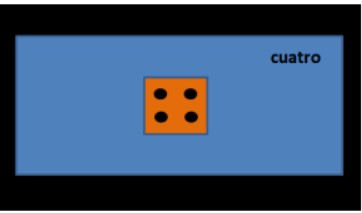
* **Dado** que soy un jugador en el juego espacial,
* **Cuando** utilizo el JoyPad para mover el Shooter,
* **Entonces** el Shooter se mueve en la dirección indicada y su nueva posición se actualiza en la pantalla.
* **Y** la cantidad de vidas del Shooter se muestra en el HUD.

**Notas Técnicas:**

* La clase GameObject debe ser abstracta y no instanciable directamente.
* Las clases Shooter y Asteroide deben implementar los métodos display() y mover().
* El HUD debe actualizarse en tiempo real para reflejar el estado actual del Shooter.

Espero que esta historia de usuario te sea útil para tu proyecto. ¿Hay algún detalle adicional que te gustaría agregar o modificar?

Punto 2: Desarrolle un videojuego que cumpla con las siguientes especificaciones: Realice un diagrama de clases



|  |  |  |
| --- | --- | --- |
|  | Fundamentos de Programación Orientada a Objetos  TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERIA  Universidad Nacional de Jujuy Trabajo Práctico N° 2 |  |

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

Mg. Ing. Ariel Alejandro Vega

|  |  |  |
| --- | --- | --- |
|  | Fundamentos de Programación Orientada a Objetos  TECNICATURA UNIVERSITARIA EN DISEÑO INTEGRAL DE VIDEOJUEGOS  FACULTAD DE INGENIERIA  Universidad Nacional de Jujuy Trabajo Práctico N° 2 |  |

balas que serán almacenadas en una lista de balas. Las balas se deben destruir cuando salen de pantalla.



Figura 1. Modelo juego Fogger

Mg. Ing. Ariel Alejandro Vega