Santiago Calle Rodas - 000290117

Conceptualizando primer juego.

1. Un sprite es una representación visual del objeto creado en el juego, es solo una imagen visual, pero cuando se crean varias y se superponen pueden generar una animación (Movimiento). En cambio el objeto es una entidad que tiene un comportamiento (evento) donde se le puede poner un sprite como representación grafica del objeto, estas entidades son la mayoría del juego, personajes, muros, paredes bolas, pero para que se vean como tales se les representa con un sprite.
2. Si, el objeto es una entidad que puede ser representada por múltiples sprites, ejemplo, un objeto que representa un personaje, el personaje lo vamos a poner a caminar, entonces hay que hacer varios sprite para que el personaje simule el movimiento de caminar, el personaje también muere, otro sprite con el personaje muerto, si ponemos a saltar el personaje hay que hacer otro sprite con el personaje saltando, todos estos sprites están en el objeto pero mediante los eventos van cambiando.
3. El asset es los archivos que tiene el juego que no son sprite, es decir, los assets pueden ser las imágenes de fondo, los sonidos del juego, efectos, niveles etc. Es cualquier dato que se pueda ingresar en el motor de juego en este caso game maker.
4. Los eventos son esos momentos del juego que pasan de acuerdo a cómo los hayas programado en los objetos. Estos comienzan desde el primer momento del juego hasta el último, y el programador puede elegir también en qué momento quiere que ocurran.
5. Create: Este evento es la primera cosa que ocurre dentro de una instancia que se da lugar en un “room”, y puede ser usado para cosas que generalmente pasan solo una vez o cuando una instancia aparece primero en el juego.

Collision: Es muy importante saber cuándo dos o más instancias de un objeto tienen que colisionar, y es un evento el cual se suelta en un objeto y se debe especificar con cuál otro objeto va a colisionar.

Mouse: El Mouse event está determinado por una serie de sub-eventos los cuales dan más precisión y facilitan a la hora de estar creando el juego, ya sea manteniendo presionado algún botón o haciendo un clic…

1. Las acciones pueden determinarse por comandos y funciones que facilitan la manipulación del juego, y vienen en conjuntos los cuales se pueden arrastrar y soltar en un evento del objeto para crear algún tipo de comportamiento en este. A esto se le llama comúnmente DnD (drag’n’drop).
2. Se podría decir que los eventos son como algo más general que las acciones, ya que las acciones se pueden arrastrar hasta los eventos para generar algunos comportamientos en un objeto. Un ejemplo de ello podría ser que el evento sea presionar la tecla izquierda del mouse, y la acción, sería que el objeto se comenzara a mover en cualquier dirección.
3. Cuando se tiene un objeto, con eventos definidos, ya se puede colocar en el “room”, y al ponerlo allí, se crea como una copia de este objeto, a lo cual se le llama instancia; y se puede poner más de una allí, y editar la posición, o el color de alguna si se quiere.
4. La diferencia es que un objeto es creado con anterioridad, al cual se le añaden los eventos; y por consiguiente, al crear una instancia, estamos obviamente utilizando un objeto, pero la instancia es la que va puesta ya en el “room”; y se pueden crear varias instancias las cuales se pueden editar por separado pero no los eventos, que fue lo que se hizo con anterioridad al crear el objeto, sino posición, fondo, entre otros.
5. …
6. Son las acciones que se realizan por segundo, el número de pasos por segundo se configura en cada room del juego, las configuraciones más comunes son 30 o 60 pasos por segundo. Si queremos que unos aviones salgan en determinado tiempo tenemos que crear un evento steps para que salgan en el tiempo propuesto.

Conceptualizando Maze.

1. El 32 significa los pixeles que están realizadas las imágenes de los sprites, en este caso la imagen tiene 32 pixeles de alto y 32 pixeles de ancho. Y ¿Qué es un pixel? es la menor unidad homogénea en color que forma parte de una imagen digital.
2. La máscara del osito es de 32 pixeles de ancho por 32 pixeles de alto
3. La máscara de colisión se utiliza para calcular cuando dos instancias están en colisión y pueden desencadenar un evento de colisión. En game maker se utiliza para delimitar los sprite y con la máscara de colisión crear eventos cuando se toquen dos objetos.
4. El origen del sprite es el punto en el que el sprite tiene una posición en el Room, cuando se crea una instancia en el room por defecto se pone en la esquina superior izquierda de la imagen por lo tanto este punto es donde se inicia la imagen en el room, game maker tiene para centrar el origen del sprite, para que el sprite cuando se ubique en el room este centrado con respecto a la habitación y no en la parte superior izquierda.
5. Cuando el solid se vuelve verdadero, se genera un evento especial de colisión con lo que la instancia se vuelve automáticamente a la posición que estaba en el paso anterior a la colisión, si se establece en falso, todas las colisiones deben ser tratadas a través del evento de colisión.
6. El muro que se implementa en solido no necesita más eventos por que los eventos se le ponen en los demás objetos, las colisiones con el muro están implementadas en los otros objetos. Lo único es que el muro es sólido para que los otros objetos puedan colisionar con el muro.
7. Es muy necesario las acciones del obj\_goal, primero porque es el objetivo del mundo en el maze, es decir, el personaje debe ir hacia la meta para ganar el mundo, segundo si no tiene acciones el personaje lo traspasa porque no es sólido y no pasa nada, pero si se le pone la acción de pasar al siguiente nivel pero no se configuro el nivel, sale un error ya que termino el nivel pero no hay un nivel siguiente.
8. La velocidad es una variable de objeto que representa la velocidad del objeto, se utiliza en combinación con la dirección.

La velocidad de un objeto se mide en pixeles por paso, cuando se configura con un valor negativo la instancia se mueve en la dirección opuesta a la dirección.

1. Las dimensiones de la grid son snap hor: 32 y snap vert: 32
2. Cuando las grid no está alineada con la persona, esta no puede pasar por los huecos adecuadamente, se puede trabar con los muros, el personaje nunca frena en la celda adecuada para pasar para otro pasillo del laberinto, el movimiento se vuelve más difícil.
3. Cuando la grid no está alineada hay que parar en la posición exacta de la grid para que el personaje pase para el otro laberinto, se vuelve imposible escapara de los enemigos ya que es difícil entrar entre dos muros.
4. Este comportamiento se debe a la máscara de colisión y la grid, la máscara de colisión es de 32x32 entonces cuando el intenta pasar por la grid de 32 y esta no está alineada perfectamente no pasa fácil, es necesario que este muy preciso para que pueda pasar por un hueco que cabe perfectamente el personaje pero sin equivocaciones, cuando la grid no está alineada es muy difícil pasar ya que es necesario un precisión muy buena para que el personaje pasa por la grid.
5. Este evento ocurre cuando los diamantes del nivel llegan a cero, cuando esto pasa se destruye la instancia que está tapando la entrada y suena un sonido (Puerta abriéndose).
6. Esta acción consiste en destruir un muro que está cubriendo la entrada, la acción lo que realiza es eliminar la instancia que está bloqueando la entrada, en este caso un bloque.
7. Se colocaron más diamantes y sin importar el número de diamantes si se vuelve cero la instancia se abre el bloque, pero en una ocasión se quedó un diamante metido con la puerta y fue imposible abrir el bloque para cogerlo
8. Cada objeto puede tener un padre, esto es como una jerarquía, entonces cuando un objeto es padre puede compartir código, acciones y eventos con ese padre. Esto se conoce como herencia y el objeto que tiene padre se llama hijo. Esto quiere decir, que se programa un padre y el resto que se cree se coloca como parent en gamemaker y este nuevo objeto coge todas las acciones del padre. Esto ayuda a ahorrar mucho tiempo a la hora de programar.
9. Con parent es muy fácil se crea un objeto que va a ser el padre, este se le implementan todos los eventos pertinentes que quiere que realice el Monter, ya cuando se configure perfectamente el padre (Monter principal) se crean los otros 5 Monter como elemento, lo único es cambiarle los sprites a estos Monter y ponerle el checbox de parent en game maker, cada objeto queda con las acciones del Monter principal pero con diferentes sprites.
10. El parent crea un padre de objeto pero los hijos que tienen las mismas acciones del padre pueden crear nuevos eventos para realizar acciones diferentes, estas acciones nuevas se sobrescriben a las acciones del padre, es decir, en el código las acciones que se sobrescriben se realizan antes que las acciones del padre.
11. El objeto controlador se coloca en el room para realizar los eventos planteados en un objeto, este se pone y las acciones del objeto se ejecutan. Es una forma para poner acciones como cambiar de room, Si se quiere poner las vidas etc.
12. Personalizar
13. Esta propiedad se caracteriza por dar profundidad a los objetos del juego, en el room se le puede asignar una deph para estar más encima de unos objetos o al contrario si queremos que un objeto este al final o lo más profundo del room, así se puede jugar como si fueran diferente capas para identificar cual esta primera y cual va de ultima
14. Personalizar: Cuando se cambia la deph de 10 a -10 la bomba sale por encima del Monter, esto se ve raro porque el Monter está caminando y cuando pasa por la bomba desaparece y vuelve a aparecer después de la bomba, esto se debe a la profundidad de la bomba que está por encima del Monter
15. Se utiliza esta función para cambiar una instancia de un objeto en otra instancia de un objeto diferente, y mientras lo hace decidir si realizar las instancias iniciales destruyen evento y las nuevas instancias crear evento. Por ejemplo, un cambio de bomba en una explosión puede destruir instancias y a la vez se puede crear otra.
16. Perform events es una función que realizara un evento especifico, para que la instancia ejecute el código. por ejemplo, si usted elige para realizar un teclado pulse evento, el evento se activará pero tecla correspondiente no será reconocido por haber sido presionado. O si se realiza un evento de alarma, el contador de alarma no se establece en -1, sino más bien seguir la cuenta atrás.
17. La opción relative en la acción obj explosión es para ver el valor exacto que tiene la explosión en el juego, se le adiciona más valor o se le reduce valor.
18. El hspeed y vspeed en la sección blocks and holes significa que el mira si la casilla de adelante está vacía, si esta vacía el bloque se puede mover hacia esa dirección, si el espacio no está vacío el bloque no se puede mover en la dirección

Conceptualizando Shooter.

1. variables es la unidad básica para la mayoría de las operaciones de programación. Las variables se utilizan para almacenar información en la memoria de los dispositivos para después poder usar la información. A las variables se le da un nombre para guardar la información puede ser desde un numero hasta una cadena de caracteres.
2. Hay cuatro categorías de variables en game maker, Cada una categoría tiene su scope:

• Instance: éstas son las variables más comunes y se definen dentro de una instancia. Es única instancia y se pueden utilizar en cualquier caso y en cualquier función dentro de esa instancia.

• Local: estas variables deben ser declaradas usando la función "var". Una variable local sólo es válida para el evento o secuencia de comandos de recursos en la que se crea.

• Global: una variable global es aquella que pertenece a la misma y no mundo del juego a cualquier instancia. Tiene que ser declaradas como globales principio, pero después de que cualquier instancia puede comprobar o cambiar la variable y el valor siempre reflejará la última operación hecha en él, no importa qué instancia o cuadro de código se ha realizado la operación.

• Built in variables: son variables especiales que se "construyen en" los objetos y las habitaciones en el mundo del juego y que puede ser única instancia o de alcance mundial (pero nunca local).

1. Está en la pregunta 2
2. Está en la pregunta 2
3. Se utiliza la variable instance que es la más común y es una instancia que se puede utilizar por cualquier evento
4. Se utiliza la variable local para utilizar eventos de recursos en la que se crea el mismo
5. La variable que se utiliza es la variable global que puede acceder a información desde varias instancias
6. Lives, Score, Health, Coins, money
7. Balas, Enemigos, Vidas, Monedas, Puntos, Velocidad, Fuerza de ataque, Armadura.
8. Vidas y vidas, estas dos variables no son iguales, una variable reconoce la mayúscula y la minúscula, entonces estas palabras como variable son diferentes.
9. La posición donde sale el obj\_bullet es 0,0 por esta razón es necesario relative, para que las balas salgan desde el obj\_myplane, con relative ya las balas se ubican en el centro del avión y salen correctamente.
10. Personalizar.
11. Esta cifra es un poco grande pero es para que las islas no se sobrexpongan de los demás objetos, es decir, que las islas van al fondo del room y los demás elementos encima de estas y así se asegura de que las islas siempre están abajo en la profundidad el room.
12. Esta función random es para que las islas salgan en diferente parte del room en el eje horizontal, si no se pone random la isla siempre sale en el mismo lugar.
13. Cada nuevo juego sale la isla en el mismo lugar porque en el room se puso ahí, pero la instancia se activa cuando este pase por el extremo inferior del room, ahí es cuando comienza a funcionar random. Como cada nuevo juego se comienza desde cero, la instancia no se ha activado por lo tanto sale la isla donde se configuro inicialmente.
14. Esta función establece un valor aleatorio. Esta crea de verdad un valor aleatorio al azar desde el inicio del juego, no como random que es después del inicio del juego.
15. No se utiliza randomize, ya que esto hace que los errores de rastreo y depuración mucho más fácil. Si desea probar con el verdadero azar, usted debe llamar a esta función en el inicio de su juego.
16. Al inicio del juego para que sea verdadero azar
17. El código significa que el avión comprueba que si la variable x es mayor que 40, esto comprueba en las coordenadas que el avión se va a mover una pequeña distancia, si esta esta vacía y no hay nada el avión salta de posición y se mueve -4 en x y 0 en y. Ojo con relative porque si no salta a la posición 0,0.
18. Al poner el centro del avión en 32,32, la bala se creará en el centro del mismo, y es allí de donde saldrá la bala.
19. Porque si la bala no se destruye al momento de salir de la vista que se tiene, puede destruir aviones enemigos que se puedan encontrar más adelante.
20. Esto se hace con alarmas, las cuales se ajustan para que se puedan disparar dos balas cada un segundo, poniendo la mitad de la velocidad total del room, la cual sería 15.
21. Para que solo se puedan disparar tres balas por segundo se debe dividir la velocidad del room en tres, y poner que la alarma se active cada 10 frames.
22. Cada que el avión principal choca con un enemigo o una bala enemiga, se le baja la barra de vida, y cuando la barra de vida llega a cero, se le quita una vida al personaje principal y vuelve a aparecer con su barra de vida en 100.
23. En controlador del enemigo se encarga de ir poniendo a los aviones enemigos en cada momento del juego.
24. Las alarmas sirven para definir los tiempos en los que se quiere que se realice determinada acción, por ejemplo, se puede poner que un enemigo cambie de dirección cada 30 steps (lo cual sería un segundo ya que la velocidad del room será 30).
25. El parámetro subimage sirve para poner que si un sprite tiene animación, ésta sea ejecutada.
26. La bala se crea cada un segundo, y se tiene que poner 30 ya que ésta es la velocidad del room.
27. Está diciendo que al ser creada, se dirija a la posición donde se encuentra actualmente el avión principal.
28. Personalización.
29. Destruye la instancia cuando sale del room, y si no se coloca, el avión enemigo puede seguir disparando aún fuera del room.
30. Permiten que al presionar espacio, se pueda crear la instancia de la bala.
31. La vista o view es lo que se quiere mostrar en la pantalla mientras se está ejecutando el juego; puede definirse para algún objeto en específico.
32. Porque si no el room también comenzará a moverse.
33. Si se coloca la velocidad en -2 en y, quiere decir que el avión se moverá hacia arriba.
34. Se tiene que poner a seguir el objeto del avión principal y tendrá la misma velocidad de este.
35. El avión no se saldrá de la vista porque lo está siguiendo a él.
36. Con el evento step, al momento de abandonar la vista actual, se le pone la acción de que se auto destruyan.
37. Se les tiene que definir una velocidad vertical positiva para que se muevan hacia abajo, que es la dirección en la que se encentra el avión principal.
38. La view in room es considerada como un área de la habitación que se va a mostrar en la pantalla, y el port on screen es considerado como el área de la pantalla donde se va a dibujar la vista.
39. Personalización.
40. Personalización.

Conceptualizando Plataforma.

1. No ocurre nada, la bola no se puede mover.
2. Si se cambia a una forma rectangular y se pone en esta posición, la figura puede moverse normal.
3. Que si un objeto está posicionado en una instancia, la gravedad se hará cero; y de ocurrir lo contario, el objeto bajará según la gravedad puesta.
4. Porque al poner esto, se le está dando un límite a la velocidad vertical, y no seguirá incrementando excesivamente.
5. Porque con esa acción se le está diciendo que al tocar objetos sólidos, la velocidad se haga cero en el momento de la colisión.
6. Al objeto saltar en un bloque, se queda quieto ya que se anula la velocidad, y hay que ponerle cero para que al colisionar con este objeto se pueda mover.
7. Se le tiene que asignar una tecla y se le debe decir que al momento de estar tocando alguna plataforma, se haga efectiva la velocidad negativa que se le va a asignar a la acción “speed vertical”.
8. Para que el objeto vuelva a bajar, se le está diciendo que cuando esté en el aire y no esté colisionando con ningún objeto sólido, se active la gravedad puesta anteriormente.
9. - El objeto deja de moverse porque se le está diciendo que solo se mueva si no está colisionando con algo en y=1, refiriéndose a un objeto sólido que esté bajo de la bola.

* En la acción “move to contact in direction direction”, se está diciendo que el objeto se pueda mover en la dirección que se le especifique cuando esté en una plataforma.

1. Esto significa que puede ocurrir que cuando la imagen tiene un sprite animado, la máscara de colisión puede cambiar en cada paso que se da, así que se debe asegurar que el personaje tenga solamente una máscara de colisión para que no ocurra esto.
2. Es la misma que la 9.
3. Significa que el límite de las “cajas” de los sprites se tienen que poner igual en todos para evitar que el personaje se mueva extraño, o que al momento de colisionar con algo, ocurra algún error.
4. Se debe poner objetos encima de los tiles y poner que no sean visibles, y a estos son los que se les asignan los eventos.