Técnicas y Conceptos para la Programación de Videojuegos 2D

Andrés Pastorini - Alvaro Martínez

TDIVJ2D – Tecnólogo Informático

Técnicas y algoritmos para VJ2D

Objetivos

Introducir los conceptos y técnicas más importantes para programar videojuegos 2D.

Conceptos básicos

Andrés Pastorini - Alvaro Martínez

Técnicas y algoritmos para VJ2D

Screen buffer

Es un área de la memoria de video, donde se dibuja lo que se mostrará en la pantalla.

Técnicas y algoritmos para VJ2D

Double Buffer

Técnica utilizada para evitar el efecto de parpadeo, consiste en tener *dos áreas de memoria,* **front-buffer***, que corresponde al área que se muestra actualmente en pantalla y la otra, el* **back-buffer***, que es donde se dibujan los objetos que formarán la escena final. Estas áreas de memoria deben tener las mismas dimensiones.* A el intercambio entre el **back-buffer** con el **front-buffer**, se le conoce con el nombre de **Page Flipping.**

Técnicas y algoritmos para VJ2D

Triple Buffer

El procedimiento de double buffer tiene como inconveniente que el cambio de *buffer* ocurre sin aviso previo, así como la exhibición de la imagen que esperaba en el *buffer fronta*l.

Cuando el procesador termina de construir la próxima imagen en el ***back buffer***, la imagen que esperaba para ser exhibida será inmediatamente enviada para la pantalla, resultando en un efecto bastante desagradable: el **screen tearing**, que son imágenes sobrepuestas.

Técnicas y algoritmos para VJ2D

Triple Buffer

Ese efecto ocurre por la falta de sincronización entre el procesador gráfico y el monitor, lo que provoca que se exhiban imágenes incompletas. En un primer momento, se puede resolver este problema “forzando” una sincronización entre el monitor y el procesador, este procedimiento es llamado “sincronización vertical”(vsync).

Con el objetivo de evitar el efecto **tearing** y también la lentitud en el procesamiento de imágenes (**double *buffer* con vsync**), se añadió un tercero espacio de almacenamiento temporal (Triple Buffer).

El *buffer fronta*l sólo se vacía de acuerdo con la actualización del monitor.

Técnicas y algoritmos para VJ2D

Triple Buffer

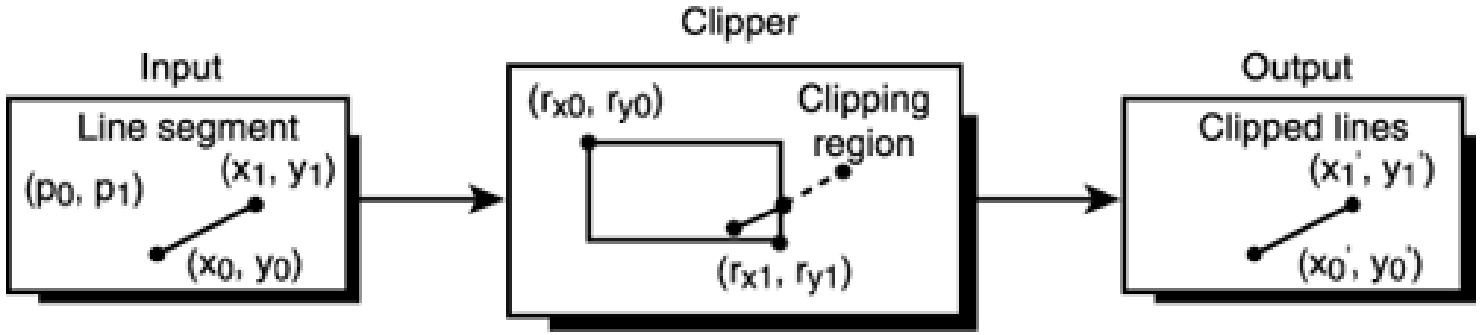
El procesador gráfico no estará más limitado por la velocidad de actualización del monitor. Cuando haya terminado una imagen para exhibir, comenzará a trabajar en la próxima, que será almacenada en el ***back buffer***. Así, el procesador puede trabajar libremente a su máxima velocidad — ya que no depende del vaciamiento de un sólo *buffer* de apoyo para comenzar el procesamiento de la próxima imagen.

El software se mantiene “dibujado” todo el tiempo “por detrás”.

Técnicas y algoritmos para VJ2D

Clipping

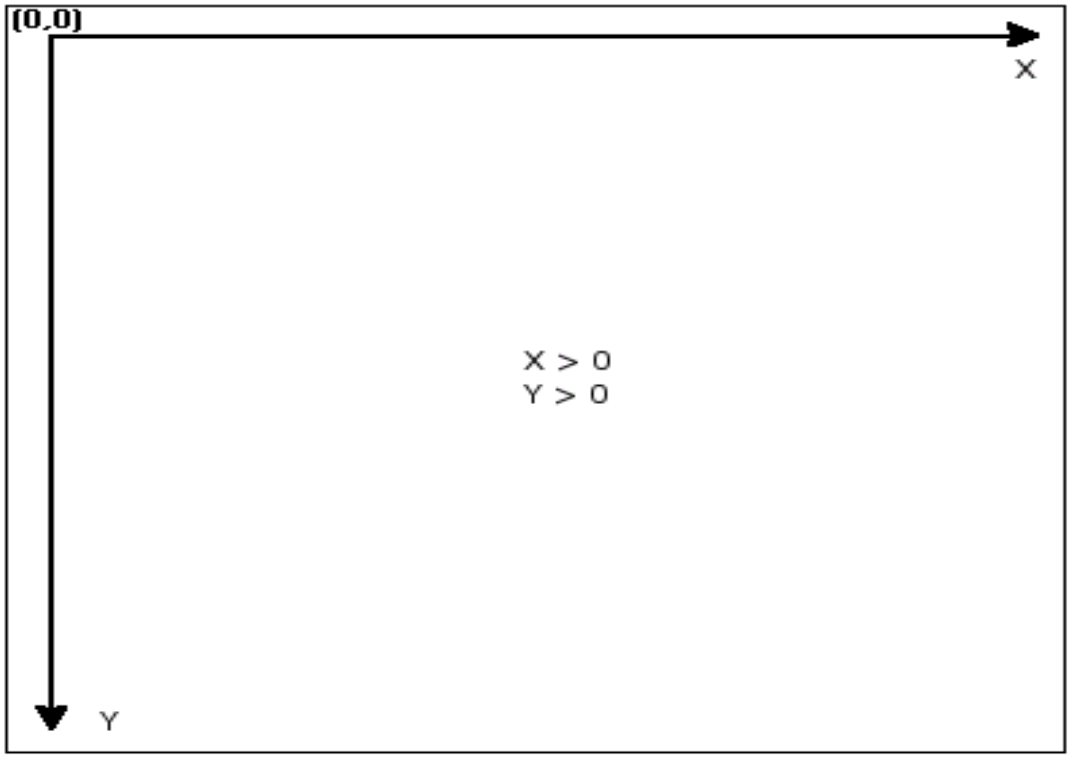
Consiste en definir una área de recorte para la pantalla, todo lo que se dibuje fuera de esta área no se mostrará en pantalla. Normalmente esta área de recorte coincide con la resolución elegida pero no necesariamente.



Técnicas y algoritmos para VJ2D

Sistema de coordenadas

Cuando dibujamos algo en pantalla, siempre se tiene que informar la posición en la cual se dibujará el objeto, y el tipo de coordenadas que se manejan(enteras, reales).



Técnicas y algoritmos para VJ2D

Sincronización

Busca que todos los objetos del juego se muevan a una determinada velocidad independientemente del computador donde se ejecute.

Existen dos métodos:

- **Sincronización por framerate**
- **Sincronización por tiempos**

Técnicas y algoritmos para VJ2D

Sincronización por Framerate

Cuando comienza el ciclo del videojuego, se obtiene el tiempo transcurrido hasta ese momento.
Luego se procesa la entrada, IA, lógica del juego, detección de colisiones, dibujo de gráficos, etc. y antes de terminar el ciclo, se obtiene el tiempo transcurrido hasta ese momento, se calcula la diferencia de tiempos y se verifica si es menor al FPS pre establecido. En caso de ser menor duerme hasta que pase el tiempo faltante.

De esta forma cada vez que se ejecute el programa, éste esperará el tiempo adecuado para cumplir con la cantidad de FPS preestablecidos, independientemente del HW donde se este ejecutando.

Técnicas y algoritmos para VJ2D

Sincronización por Tiempo

Este método no se guía por el **framerate**, busca que los objetos se muevan a la misma velocidad en todas las máquinas donde se ejecute.

Este método se usa bastante en video juegos 3D, ya que el **framerate** varía mucho en cada ciclo, dependiendo de la cantidad de objetos que se deban **renderizar**.

Tiene como desventaja que a pesar de que los objetos se mueven siempre a la misma velocidad, en un computador más lento, el desplazamiento puede no verse fluidamente.

Técnicas y algoritmos para VJ2D

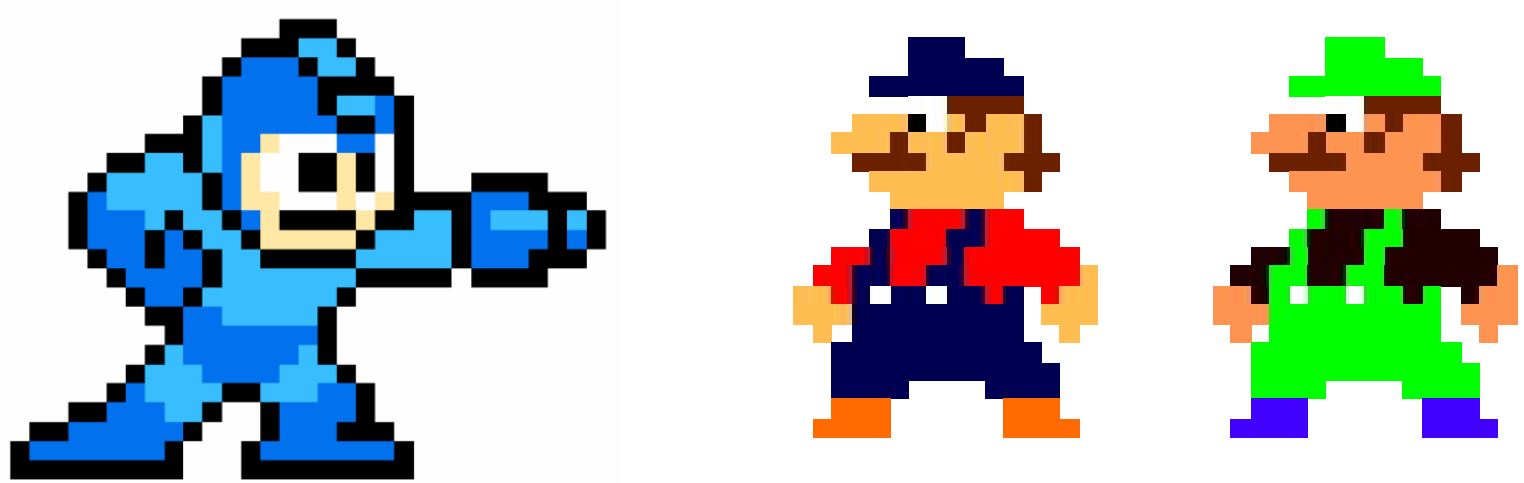
Sincronización por Tiempo

La técnica consiste en calcular la posición de un objeto obteniendo el tiempo que ha transcurrido hasta el momento de calcular la nueva posición del objeto, y luego se multiplica ese tiempo por la velocidad con la cual queremos que funcione el video juego.

Técnicas y algoritmos para VJ2D

Sprite

Es cualquier objeto que aparece en el videojuego. Normalmente tiene asociados atributos como una imagen, un conjunto de animaciones, una posición, etc.



Técnicas y algoritmos para VJ2D

Sprite

Pueden ser estáticos o dinámicos.

- *Los estáticos son los que no cambiarán su posición durante el desarrollo del juego.*
- *Los dinámicos, son los que tienen algún tipo de movimiento.*

Se representan en memoria normalmente con algún tipo de estructura.

Técnicas y algoritmos para VJ2D

Animación de Sprites

Son una secuencia de imágenes que son dibujadas rápidamente, que dan la sensación de movimiento fluido.

Un **frame** corresponde a una de las imágenes que forman parte de la animación de un **sprite**.

Técnicas y algoritmos para VJ2D

Animación de Sprites

El número de **frames** dibujados en un periodo de tiempo se conoce como **frame rate**.

Si cada uno de los **frames** se muestra muy rápido o muy lento, la ilusión de movimiento se pierde y el usuario verá cada uno de los **frames** como una imagen separada.

El valor del **frame rate,** para ser percibido por el ojo humano como movimiento continuo debe ser superior a 25 veces por segundo.

Técnicas y algoritmos para VJ2D

Animación de Sprites

Las animaciones pueden ser externas, internas, o una combinación de ambas.

Una animación externa es equivalente a una traslación del **sprite** a lo largo de la pantalla.

Una animación interna está compuesta de un número determinado de fotogramas o imágenes que definen el comportamiento del personaje.

Técnicas y algoritmos para VJ2D

Animación de Sprites

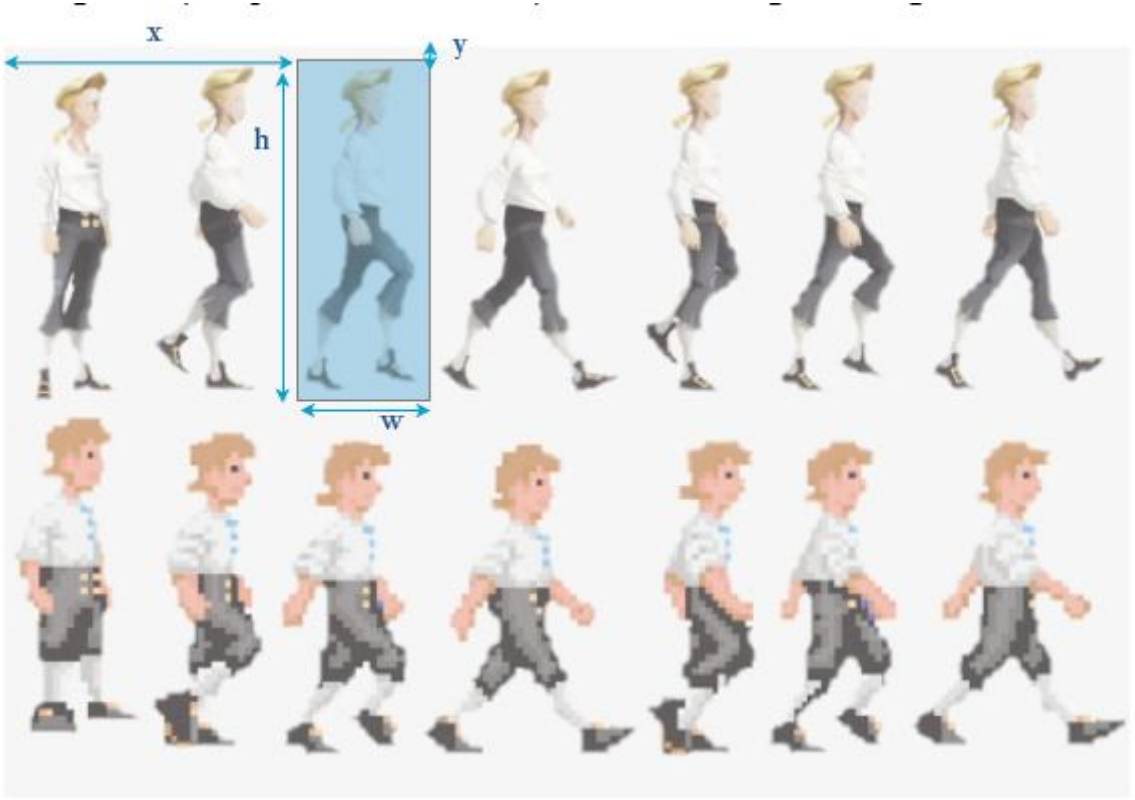
Las animaciones se representan internamente con algún tipo de estructura de datos, como pueden ser un vector o una lista.

Normalmente el **sprite** tendrá como atributo una lista de animaciones, donde cada animación tendrá otra lista con cada uno de los **frames**.

La totalidad de los **frames** que forman un personaje suele almacenarse en un solo archivo de imagen(**sprite sheet**), que es una imagen que contiene las secuencias de **frames** para las diferentes acciones que puede realizar un personaje.

Técnicas y algoritmos para VJ2D

Animación de Sprites



Técnicas y algoritmos para VJ2D

Animación de Sprites

A la hora de implementar la animación se debe tener en cuenta la velocidad de ejecución y los momentos en que aparezcan los distintos **frames** de esta, para que se produzca un efecto visual realista.

La estructura que representa a la animación debe incluir, entre otros atributos, el **frame rate** de ejecución y debe ser consistente con el estado del personaje.

Técnicas y algoritmos para VJ2D

Transparencias - Color keys

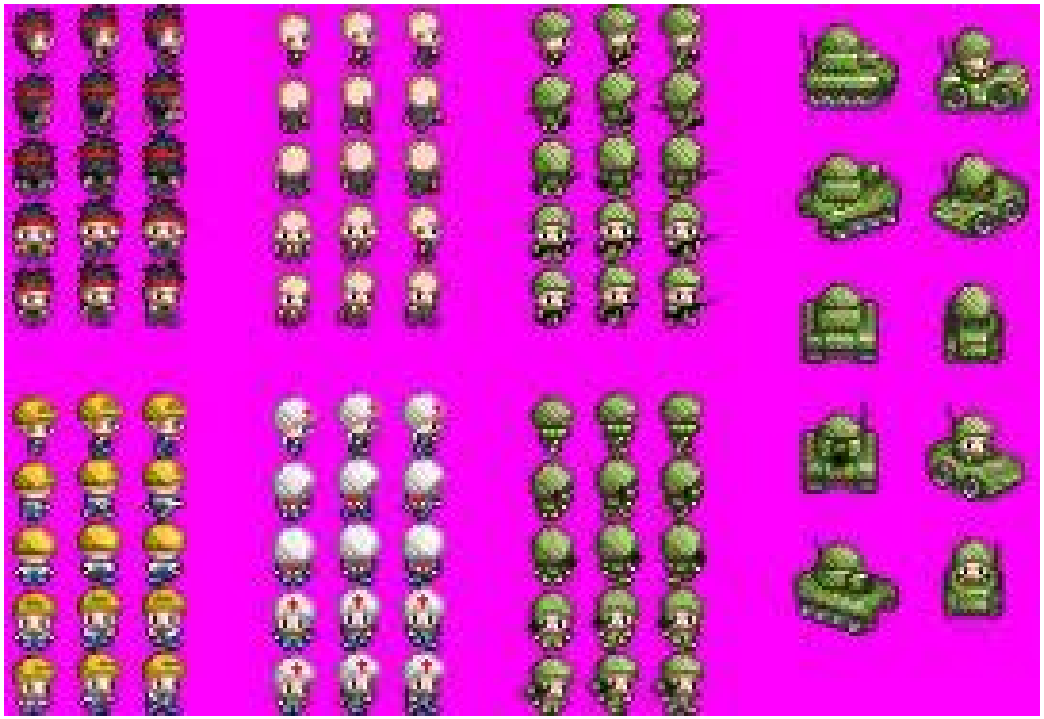
Cuando se visualiza una imagen en pantalla, esta siempre estará en un formato con forma rectangular, que dentro tendrá la representación de un personaje, objeto, etc. De esta imagen se puede querer solo visualizar la forma que representa.

Para esto, la imagen debe estar almacenada en un tipo de formato que permita transparencias, sino se debe elegir un color que sea interpretado como transparente, pintar con ese color las zonas que no se quiere que aparezcan en pantalla y luego dibujar la imagen ignorando dicho color.

Normalmente se utiliza el color magenta (255, 0, 255), como color de transparencia, ya que es poco probable que se encuentre en una imagen.

Técnicas y algoritmos para VJ2D

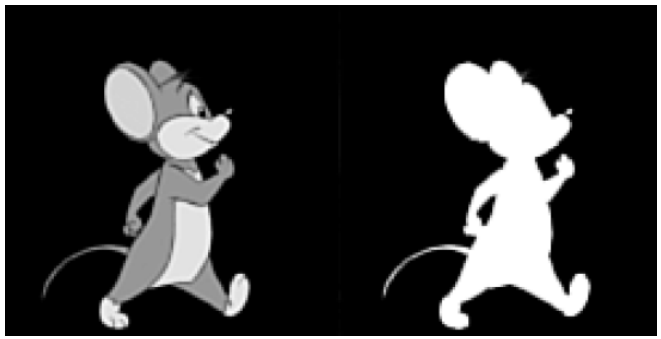
Transparencias - Color keys



Técnicas y algoritmos para VJ2D

Transparencias - Máscara

Existe otro método para ignorar un color en una imagen, y es que se puede usar una máscara que se antepone a la imagen original, indicando cuales pixeles de la imagen se quieren que sean transparentes. El color utilizado para indicar transparencia es el negro, y el color blanco indica las zonas visibles. Al dibujar la imagen, se recorre cada pixel de la máscara, y si este color es blanco, se muestra el pixel de la imagen original en esa posición.



Técnicas y algoritmos para VJ2D

Transparencias

Canal Alpha

Es una técnica para transparentar una imagen. Para lograr esto se agrega un cuarto canal llamado canal **alpha,** los colores se representan como RGBA. Cada valor alfa de un pixel representa su grado de opacidad, donde un valor cero indica un 100% de transparencia, y un valor 255 indica que es totalmente opaco.

Surface.convert_alpha(Surface): return Surface
Surface.convert_alpha(): return Surface

Técnicas y algoritmos para VJ2D

Transformaciones

A un **sprite** en pantalla, es posible aplicarle algunas transformaciones geométricas como son

- Traslación
- Rotación
- Escalado
- Simetría(flip)

Técnicas y algoritmos para VJ2D

Transformaciones

Traslación

Cambia la posición del **sprite** para luego dibujarla en otro lugar de la pantalla, dando la sensación de movimiento.

Se puede tanto utilizar para mover a un elemento o para crear una animación externa.

Introducción a Python y PyGame

Transformaciones - Rotación

Gira un **sprite** un número determinado de grados.

pygame.transform.rotate(Surface, angle): return Surface

Técnicas y algoritmos para VJ2D

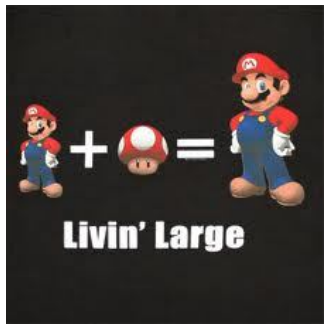
Transformaciones - Escalado

Cambia el tamaño del **sprite** según un **factor de escala**.

Si se aumenta el tamaño, se produce un efecto no deseado en los bordes de la imagen, conocido como **aliasing**, es decir, las líneas o curvas que forman los bordes de la imagen se ven discontinuas.

Para contrarrestar esto existe el **antialiasing**, que es una técnica que permite suavizar todos los bordes y así disminuir el efecto.

pygame.transform.scale(Surface, (width, height), DestSurface = None): return Surface



Técnicas y algoritmos para VJ2D

Transformaciones - Flip

Aplica una transformación de tipo espejo al **sprite**, esta puede ser horizontal y vertical.

pygame.transform.flip(Surface, xbool, ybool): return Surface

Técnicas y algoritmos para VJ2D

Surface

Zona de memoria que puede estar en la RAM o en la memoria de video y es usada para almacenar una imagen.

Programáticamente se representa como una estructura o clase que tiene como mínimo las misma propiedades de un archivo de imagen.

Técnicas y algoritmos para VJ2D

Blitting

Es una operación para transferir un bloque de bytes de un sector de la memoria a otra. Es una forma de renderizar **sprites** sobre un fondo. Esta operación es una de las más críticas, y puede ser acelerada por hardware.

Se puede especificar en que posición de la superficie destino se copiará la superficie origen, incluso que parte de la superficie origen.

Técnicas y algoritmos para VJ2D

Dirty rectangles

Se copia a la memoria solo las áreas de la pantalla que han cambiado, cada vez que algún **sprite** cambie su posición, se copia a la memoria de video el área de un rectángulo que contenga al **sprite** y se colocan en las mismas coordenadas en la Video RAM.

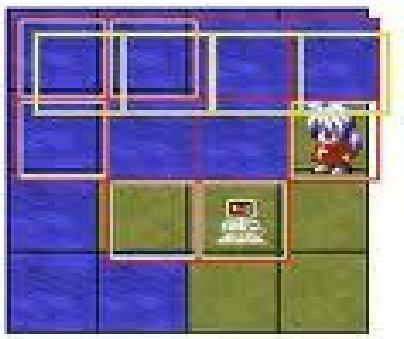
Técnicas y algoritmos para VJ2D

Tile mapping

Es una técnica muy utilizada desde los comienzos de la implementación de video juegos, y sirve para representar los mapas de cada nivel.

Puede llegar a disminuir mucho el tamaño que ocupa la/s imagen/es de un nivel del juego en la memoria.

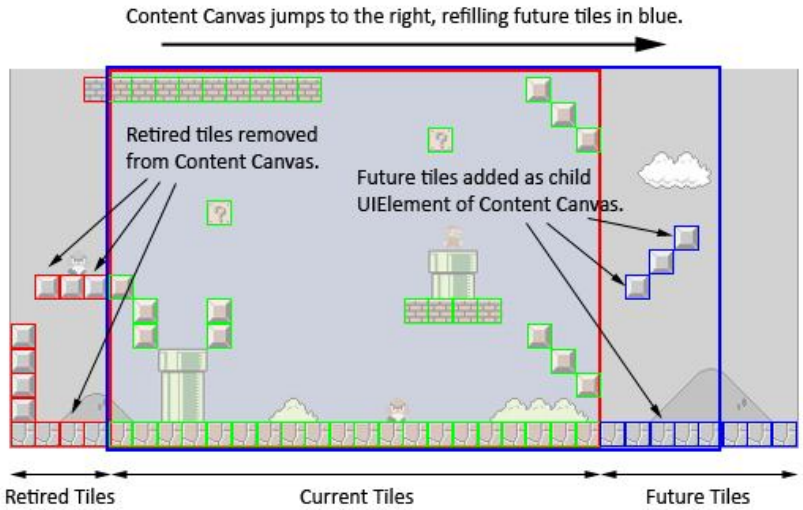
La idea principal consiste en dividir el universo del juego en un conjunto de mosaicos (“tiles”). Cada tile representa un patrón visual rectangular que se combinara con otros tiles para representar visualmente al nivel.



Técnicas y algoritmos para VJ2D

Tile mapping

Si por ejemplo en el universo del juego aparecen piedras, pasto, nieve , etc., para representar estos elementos gráficos se utilizaran tiles que se agregaran al mapa. El tamaño que ocupa en memoria la representación grafica del nivel se reduce ya que se utilizara la misma imagen para llenar los elementos del mismo tipo que aparecen en el mapa.



Técnicas y algoritmos para VJ2D

Tile mapping

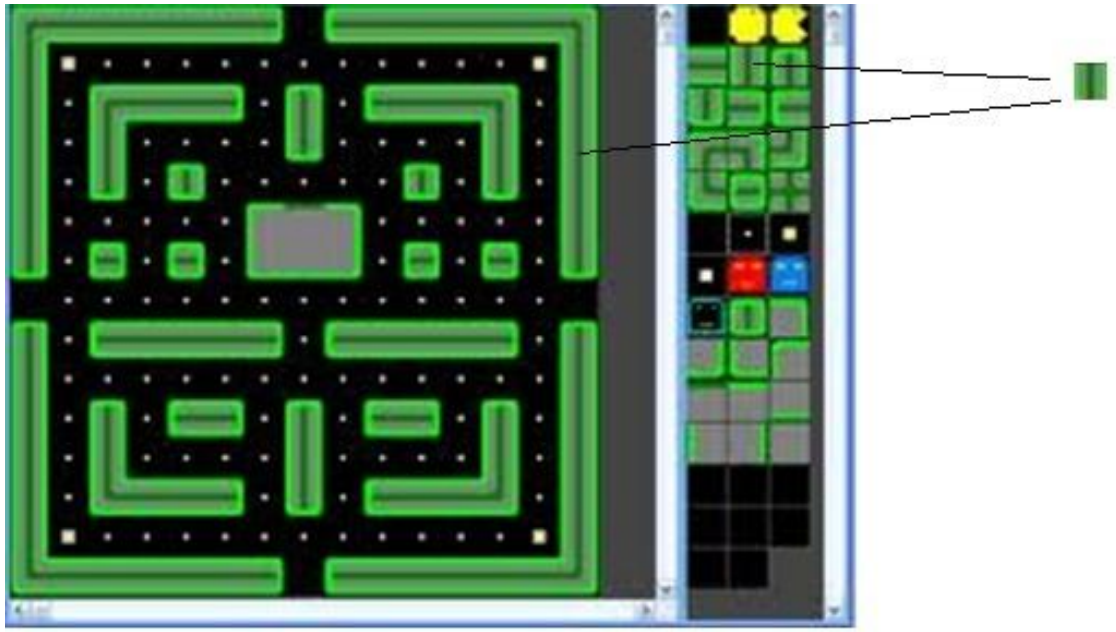
Para utilizar está técnica, se utilizan tres elementos :

- Tile
- Tile sheet (Tile Set)
- Tile Map

Técnicas y algoritmos para VJ2D

Tile

Es la unidad mínima que compone un mapa de tiles, es una imagen rectangular que sirve para componerla parte gráfica del nivel junto con los demás tiles del tile set.



Técnicas y algoritmos para VJ2D

Tile Set (o tile Sheet)

Es un archivo de imagen que contiene todos los tiles que se utilizan en el mapa. Para acceder a cada tile particular se accede a la imagen, y dentro de estas a las coordenadas del rectángulo que contiene el tile especifico.

Al diseñar un escenario, se puede diseñar éste utilizando distintas capas de imágenes, por ejemplo una capa para el pasto, otra para el suelo, otra para rocas, otra para el fondo que no es obstáculo.

Técnicas y algoritmos para VJ2D

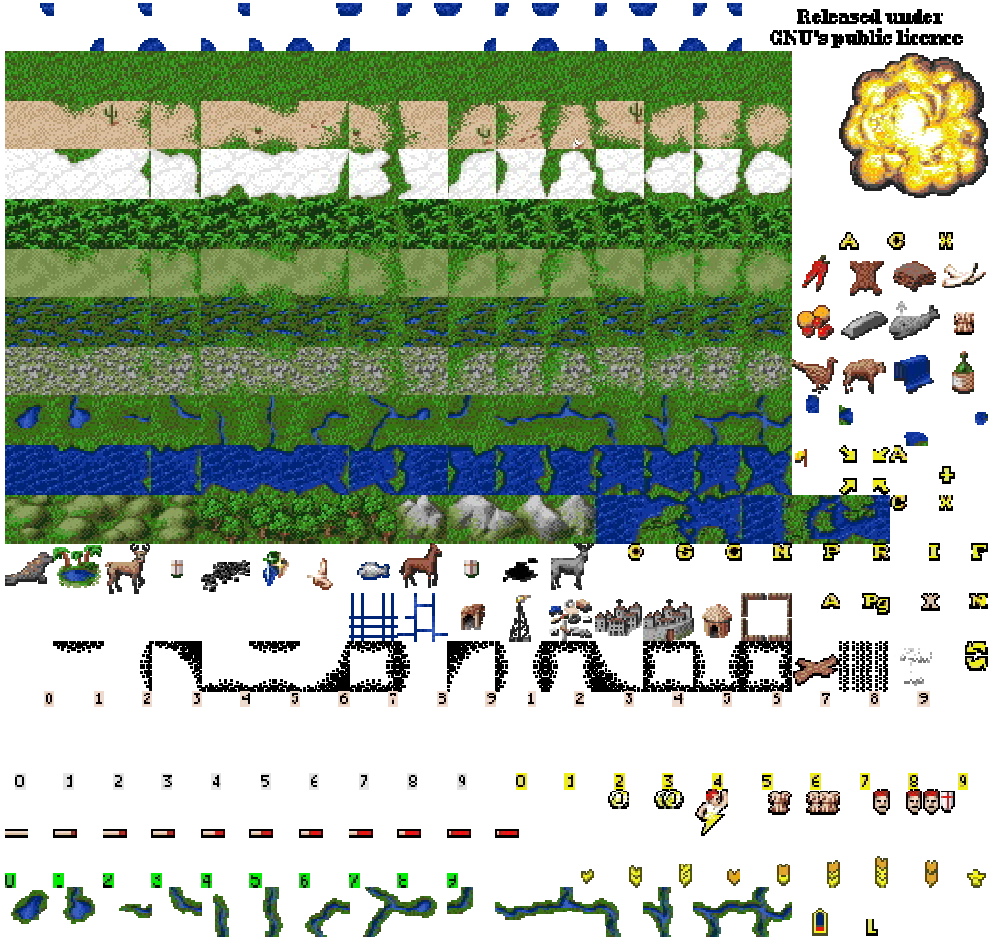
Tile Set (o tile Sheet)

Esto es útil para trabajar el diseño por partes, y para asignar distintas propiedades a cada capa(por ejemplo para indicar que los elementos de cierta capa producen colisiones o otros comportamientos contra el personaje).

Las dimensiones de cada tile en una misma capa es conveniente que sean las mismas.

Técnicas y algoritmos para VJ2D

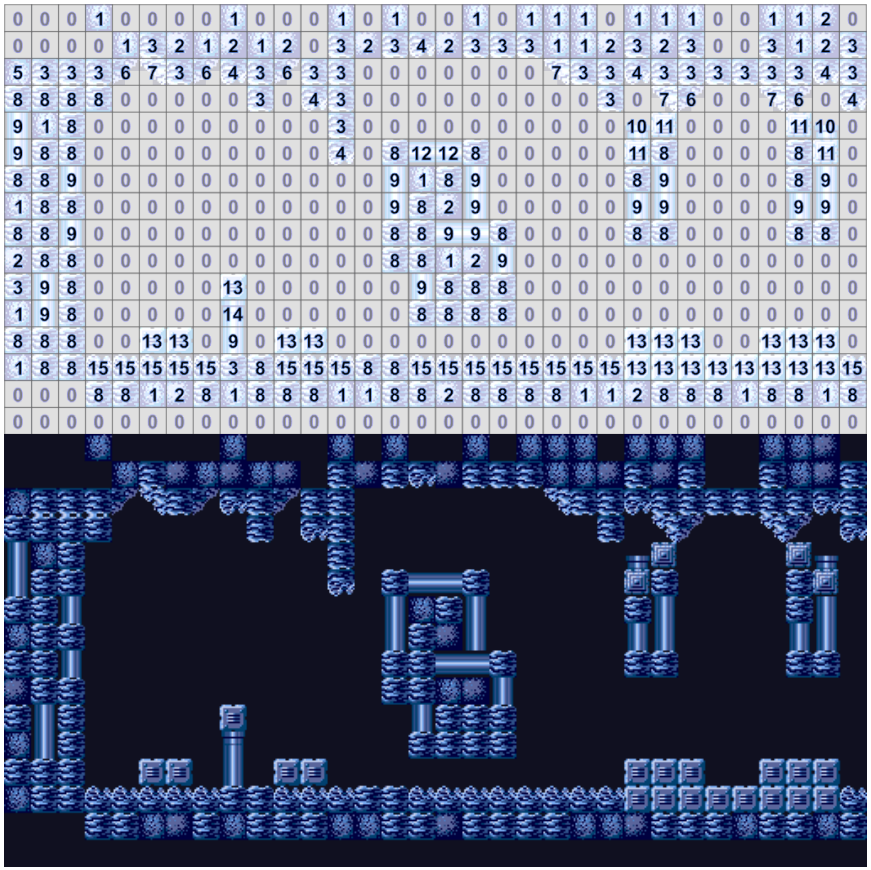
Tile Set (o tile Sheet)



Técnicas y algoritmos para VJ2D

Tile map

Es un archivo con formato texto plano, xml, etc., que indica para cada coordenada del mapa que tile se debe dibujar



Técnicas y algoritmos para VJ2D

Tile map

```
# Map file for tile-based game
# (Lines that start with '#' are comments)
# The tiles are:
#   (Space) Empty tile
#   A..Z    Tiles A through Z
#   o       Star
#   !       Music Note
#   *       Goal
#   1       Bad Guy 1 (grub)
#   2       Bad Guy 2 (fly)
AD         IIIIIII              IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
AD                      o o o                 I                 E
AD                     IIIIIII                I                 C
AD                                            I           o o   C
AD                    2  !         2                      IIIII  C
AD                      IIIII                                    C
AD         III                          1    1  1        2     C
AD                            1      IIIIIIIIIIIII              C
AD      1       EBBBBBBBBBBBBBBF     o o o o o o             * C
AHBBBBBBBBBBBBBBGAAAAAAAAAAAAAAAHBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBG
```

Técnicas y algoritmos para VJ2D

Tile map

Los tile **map** se suelen almacenar en memoria en una estructura de tipo matriz bidimensional a la cual normalmente se le llama **Matriz de mapeo**. Para cargar un tile **map** en pantalla, se recorre cada uno de sus elementos, y para cada uno se obtiene el tile correspondiente , luego se hace un **blit** en la región especifica de la superficie que representa a la pantalla.

Algoritmos

Andrés Pastorini - Alvaro Martínez

Técnicas y algoritmos para VJ2D

Screen based games
Son los videojuegos en que las dimensiones de cada nivel se pueden ver completamente en la ventana de la pantalla(juegos sin scroll). Por ejemplo pacman, donkey-kong nes,etc.

Técnicas y algoritmos para VJ2D

Scrolled Screen based games(2 y 4 way scrollers)
Se utiliza esta técnica, cuando se quiere crear un universo de juego con dimensiones mayores a las que se podrían visualizar en la ventana en la que se está jugando.

Ejemplos de juegos que utilizan esta técnica son 1942, Super Mario Bros, etc.

Técnicas y algoritmos para VJ2D

Scrolled Screen based games(2 and 4 way scrollers)
El mundo del juego es más grande que las dimensiones de la pantalla, por lo que solo se procesarán la áreas que sean visibles para el jugador, se necesita la posición de éste en la pantalla para saber lo que se debe dibujar.

Primero que nada se deberán elegir las porciones de la ventana visible, esto será como un marco que abarcara parte de todo el escenario, y será la parte que el jugador vea en determinado momento, para visualizar otras partes del escenario, esta ventana se ira trasladando en función del movimiento del jugador.

Técnicas y algoritmos para VJ2D

Scrolled Screen based games(2 and 4 way scrollers)

Técnicas y algoritmos para VJ2D

Scrolled Screen based games(2 and 4 way scrollers)

El nombre “2 AND 4 WAY SCROLLERS” se refiere a que es posible utilizar esta técnica para manejar el **scrolling** horizontal(super mario bros), vertical(elevator) o ambos(limbo, super mario bros 3).

Algoritmo:

- Obtener los tiles a mostrar en pantalla.
- Obtener las coordenadas en pantalla para cada tile.
- Dibujar cada elemento del mapa que se encuentra en el rango de tiles a dibujar.

Técnicas y algoritmos para VJ2D

Multilayered Engines

Permite codificar mapas utilizando más de una capa de tiles. **(background layer, tree layer,etc).**
Esta técnica es útil para dividir el trabajo artístico, combinar distintos tipos de tiles, mover objetos sobre el fondo, dar ilusión de profundidad, etc.

Usa varias matrices de mapeados para codificar el mapa del juego.

Una matriz podría representar el fondo, otra podría ser una capa de arboles, etc.

Técnicas y algoritmos para VJ2D

Multilayered Engines

Se asume que las capas están ordenadas desde el fondo hasta el frente en orden ascendente empezando por el id con valor cero, cuanto mayor es el valor más cercana esta la capa al observador.

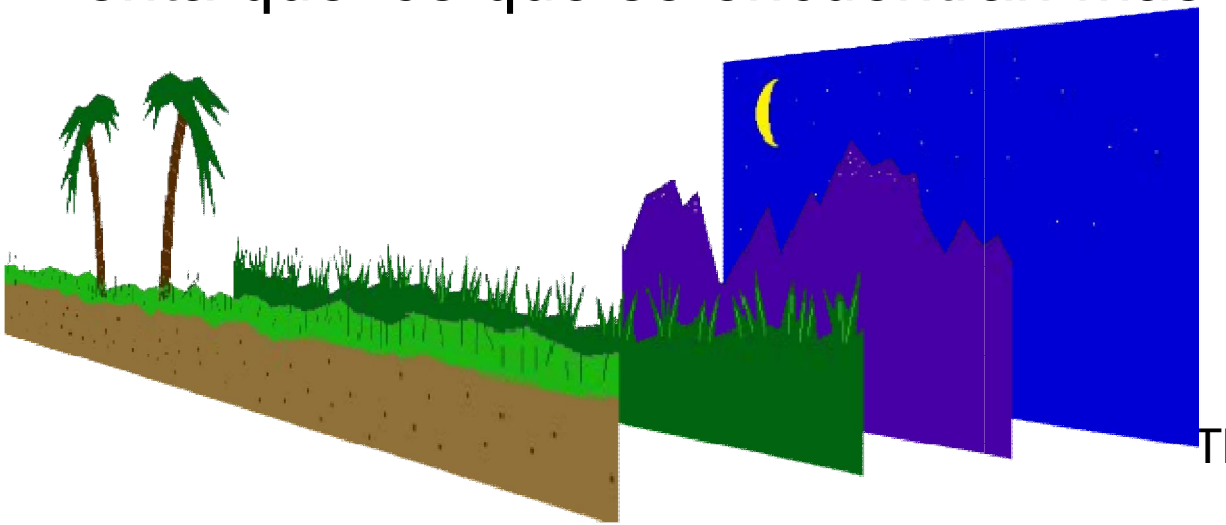
Usualmente la capa cero cubre todo el fondo de la pantalla, pero las capas más cercanas pueden tener tiles vacíos.

Por esta razón las capas superiores necesitan algún tipo de optimización para no gastar tiempo dibujando tiles vacios. Usualmente se reserva un valor que se utiliza en el tile **map** para indicar que el tile no se debe dibujar.

Técnicas y algoritmos para VJ2D

Parallax Scrolling

Esta técnica sirve para generar la sensación de profundidad y perspectiva en un escenario plano con desplazamiento(scroll). Parallax es un fenómeno óptico que describe el aparente desplazamiento de un objeto distante, cuando es visto desde dos posiciones distintas.
La aparente velocidad de los objetos distantes es más lenta que los que se encuentran más cerca.



Técnicas y algoritmos para VJ2D

Parallax Scrolling

Si se manejan un conjunto de capas de tiles que representan el fondo del escenario, las capas y por lo tanto los objetos que pertenecen a ellas, tienen una distancia asociada al observador.

De esta forma, con la distancia de cada capa, se puede simular el fenómeno de **parallax**, moviendo las capas a distintas velocidades, logrando así un efecto de profundidad.

Para lograr el efecto **parallax** visible entonces solo hay que mover las capas a distintas velocidades.

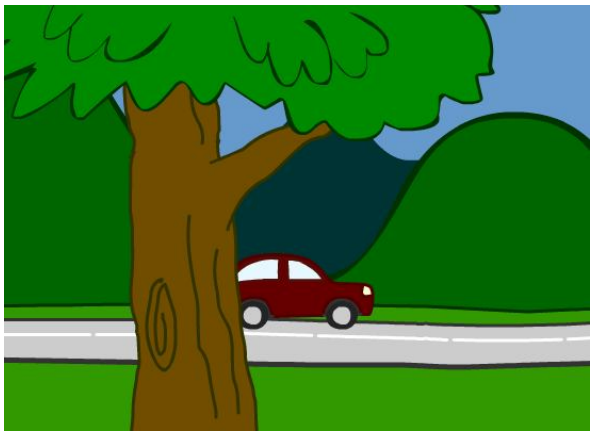
Técnicas y algoritmos para VJ2D

Parallax Scrolling

Por comodidad, a la hora de implementar al algoritmo, se ordenan todas las capas en una lista, y se ubica a la capa más lejana en la posición cero(capa que se moverá a menor velocidad), y las capas más cercanas en posiciones mayores según su cercanía.

Técnicas y algoritmos para VJ2D

Parallax Scrolling



```
def parallax:
     keys = pygame.key.get_pressed()
     if  keys[pygame.K_RIGHT]:
          for layeri in xrange(0,numlayers):
             playerx[layeri]=layeri + 1

     for layeri in xrange in (0,numlayers):
        for  yi in xrange(beginy,endy):
          for  xi in xrange(beginx,endx):
               screenx=xi*tile_wide – playerx[layeri]-screenplayerx;
               screeny=yi*tile_high – playery[layeri]-screenplayery;
               tileid=mapping_matrix [layeri][yi][xi];
               if (tileid>0) :
                    screen.blit(tile_table[tileid],(screenx,screeny))
     pygame.display.flip()
```

Técnicas y algoritmos para VJ2D

Isometric Engines

La perspectiva isométrica permite generar la sensación de un entorno 3d sobre una superficie plana. Consiste en la representación de un objeto o un nivel de juego visto de lado desde un punto de vista alto, donde todos los elementos están girados 45 grados. Se trata de un
perspectiva paralela, lo que significa que las líneas no convergen como en la proyección en perspectiva cónica. Por lo tanto, la perspectiva isométrica no sufre distorsión y mantiene las proporciones reales del objeto y sus relaciones.

Técnicas y algoritmos para VJ2D

Isometric Engines

Los juegos con perspectiva isométrica son codificados de forma muy similar a los juegos habituales 2D.

La única diferencia es que los tiles tienen forma romboidal y la rutina de proyección en pantalla es un poco más compleja.

El área exterior al tile romboide queda en blanco para que sea mas sencillo la ubicación de los demás **tiles** vecinos.

Técnicas y algoritmos para VJ2D

Isometric Engines

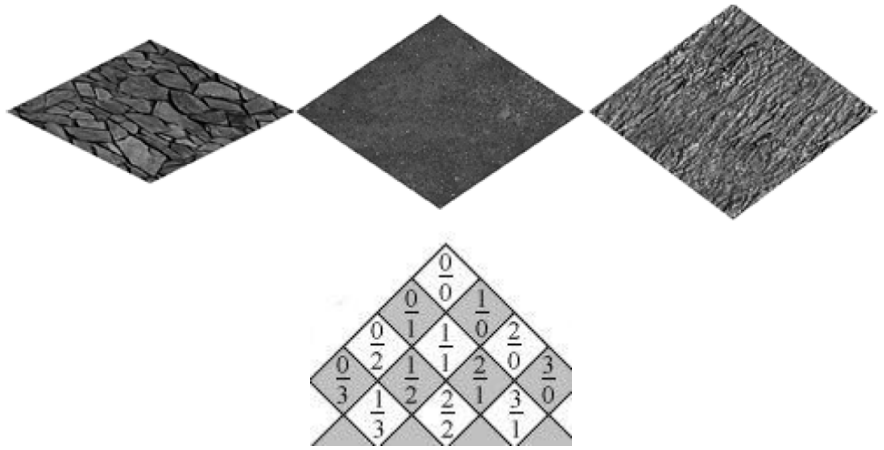
El dibujado del mapa requiere mantener una mascara de transparencia activa siempre para no dibujar la parte exterior al rombo en cada tile.

En general los tiles en esta perspectiva tienden a ser más anchos que altos, la razón de esto es que los elementos dibujados con esta perspectiva solo lucen convincentes cuando el observador no se encuentra a demasiada altura al nivel del suelo.

Técnicas y algoritmos para VJ2D

Isometric Engines

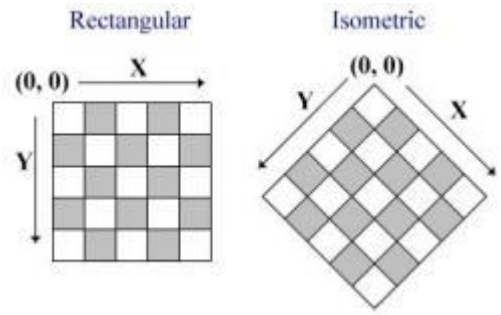
Se utilizan tiles rectangulares, pero el contenido visible tiene forma de rombo. Pero al dibujar solo se toma en cuenta la altura y el ancho. El método de dibujado sigue siendo de izquierda a derecha y de arriba abajo.

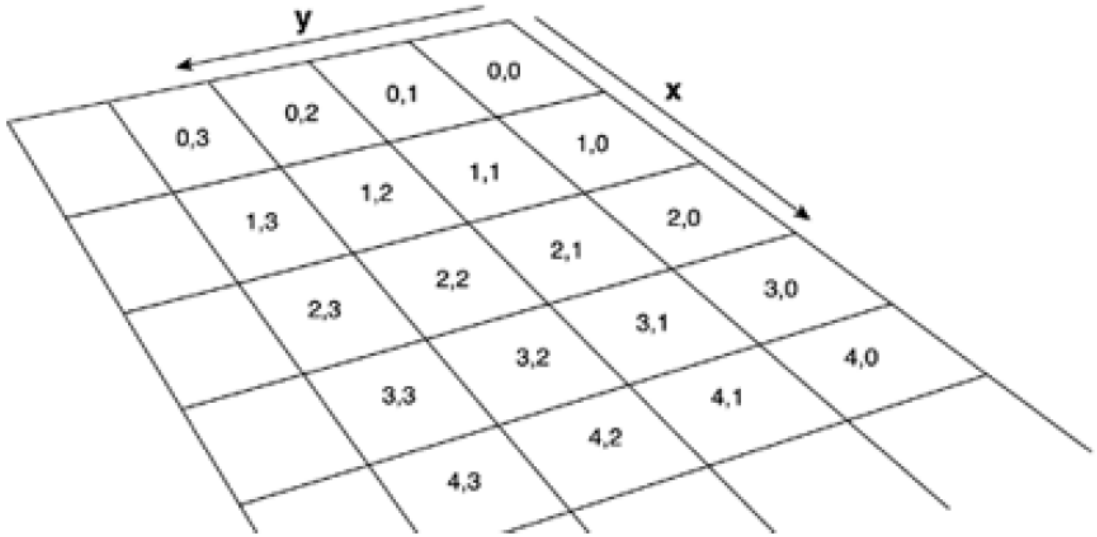


Técnicas y algoritmos para VJ2D

Isometric Engines

La representación isométrica de las baldosas se debe realizar con una rutina de proyección. Se proyecta de forma diagonal en la pantalla. Recordar que la pantalla sigue siendo rectangular.

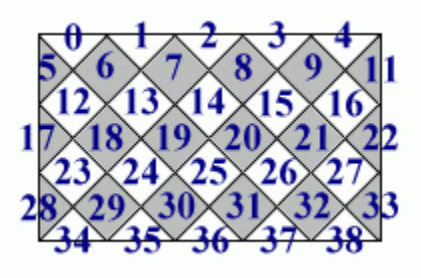


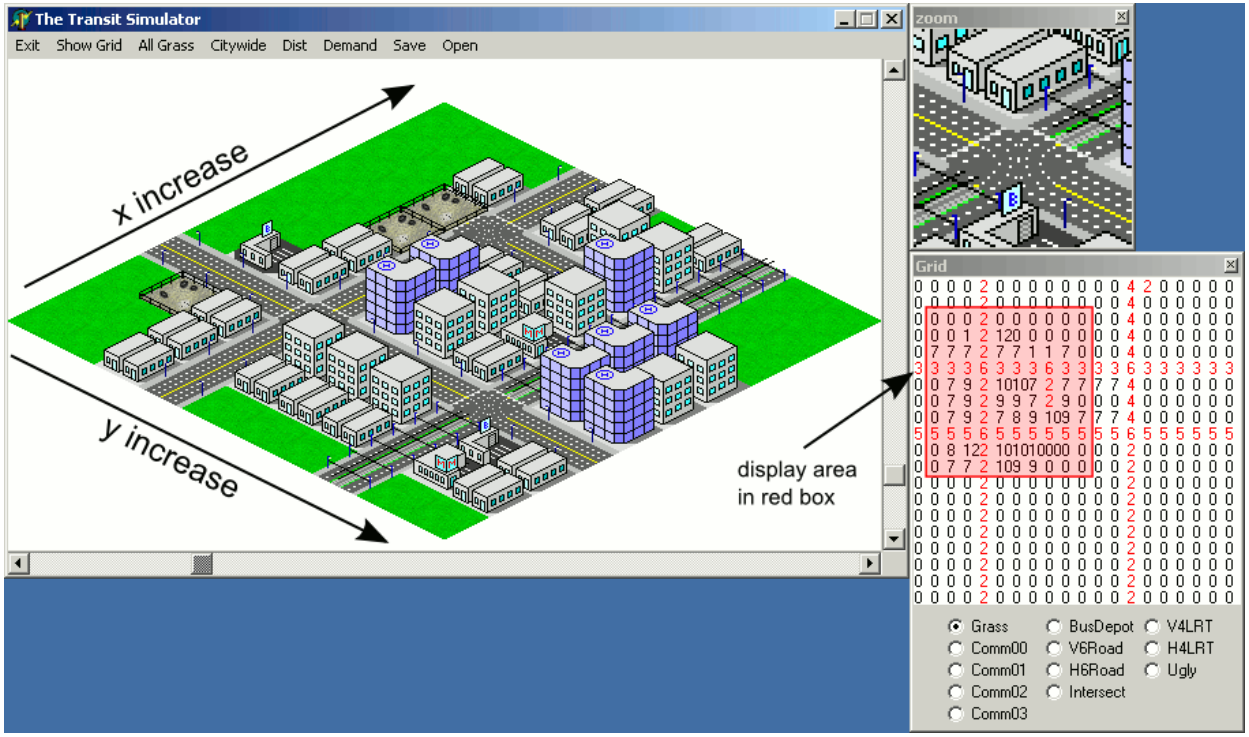


Técnicas y algoritmos para VJ2D

Isometric Engines

Orden en que los tiles son dibujados





Técnicas y algoritmos para VJ2D

Page Swap scrollers

Esta es una variante de los anteriores algoritmos de scrolling, pero sin la utilización de tiles. Se utiliza cuando se dispone de una pieza de arte cuyos elementos se encuentran todos incluidos en la misma imagen, y se quiere ofrecer la funcionalidad del desplazamiento.

Juegos que usan este tipo de técnica son por ejemplo **Monkey Island** y **baldour gate.**

Se trabaja con un mapa de nivel que consiste en una sola imagen y esta se divide en sectores.

Técnicas y algoritmos para VJ2D

Page Swap scrollers

Se utiliza la posición de el jugador para determinar que sectores se deben cargar en memoria, los restantes sectores quedan almacenados en memoria secundaria, pero se intercambiaran en memoria principal cuando sea necesario.

La estructura de datos más adecuada para representar el escenario para este tipo de juegos, es un **array** bidimensional. Este permite representar la ventana de datos visible.

Técnicas y algoritmos para VJ2D

Page Swap scrollers

Se necesitan dos atributos para mapear la ventana con las coordenadas del mundo. Las variables cornerx y cornery indican donde la esquina superior izquierda de la matriz debería ser ubicada en el mapa.

Técnicas y algoritmos para VJ2D

Page Swap scrollers

Detección colisiones

Andrés Pastorini - Alvaro Martínez

Técnicas y algoritmos para VJ2D

Detección de colisiones

La **Detección de Colisiones** es utilizada para detectar si dos objetos del juego han colisionado entre si. La detección puede ser:

- Por Área

- Pixel a Pixel

Técnicas y algoritmos para VJ2D

Detección de colisiones

Por Área

Los objetos son envueltos por un área rectangular o circular, cuando dos de estas áreas se superponen hay una colisión.

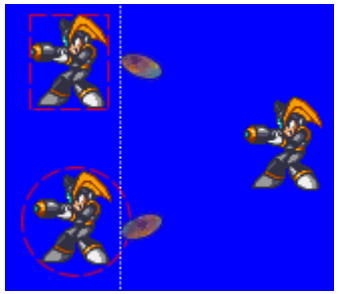
Existen dos estilos fundamentales para la detección de colisiones: la Colisión de rectángulos y Colisión de Círculos.

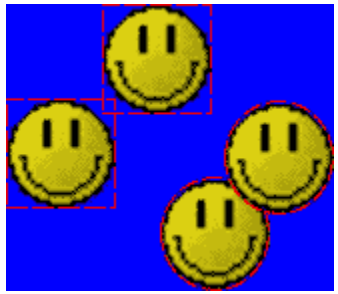
Cada cual se usa según sea conveniente de acuerdo a la figura que mejor pueda contener.

Técnicas y algoritmos para VJ2D

Detección de colisiones - Por Área

Depende principalmente de la forma del sprite, si bien todo sprite es rectangular, la figura descrita dentro de el es generalmente irregular y existen casos en los que un cuadrado no daría la precisión adecuada para la forma que contiene. Existen métodos donde las cajas envolventes están alineadas a los ejes, alineados al objeto, o se busca que sean de área mínima.

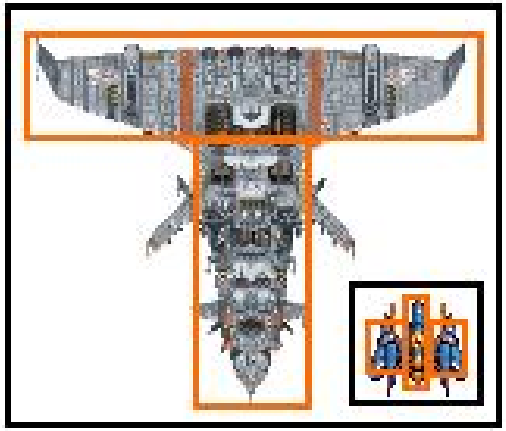




Técnicas y algoritmos para VJ2D

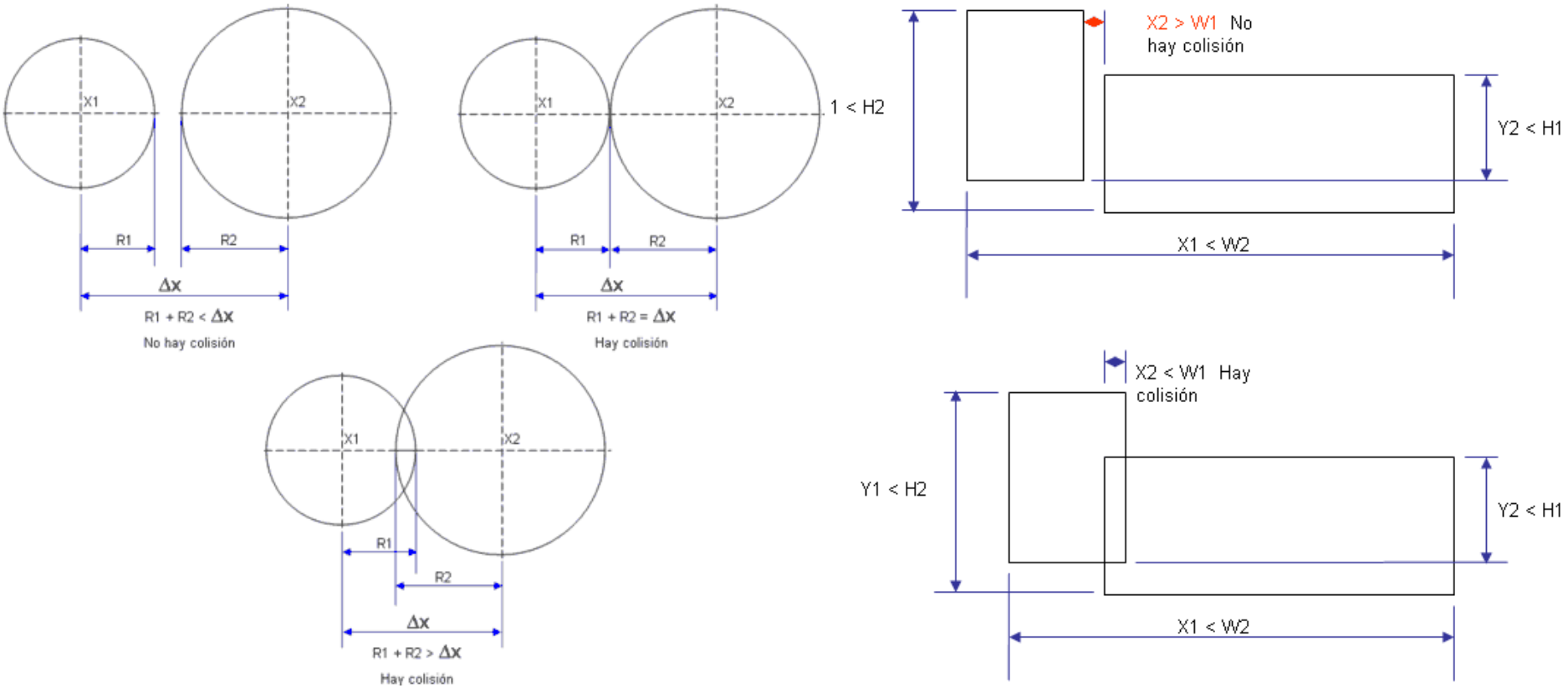
Detección de colisiones - Por Área

También es posible dividir cada uno de los **sprites** en rectángulos más pequeños, y englobarlos todos en uno más grande.
Primero se intenta detectar la colisión mediante los rectángulos grandes, y si hay colisión, se prueba detectar la colisión entre los rectángulos más pequeños (estos son los que decidirían si hay colisión o no).



Técnicas y algoritmos para VJ2D

Detección de colisiones - Por Área



Técnicas y algoritmos para VJ2D

Detección de colisiones - Pixel a Perfect

Los objetos ocupan un área rectangular, pero tienen una mascara que define que píxeles son visibles. Primero se realiza una detección de colisión de área, luego, si hubo colisión, se realiza una detección pixel a pixel entre los pixeles superpuestos de ambos objetos. Si existen dos pixeles superpuestos, y ambos son visibles, entonces hay una colisión.

Técnicas y algoritmos para VJ2D

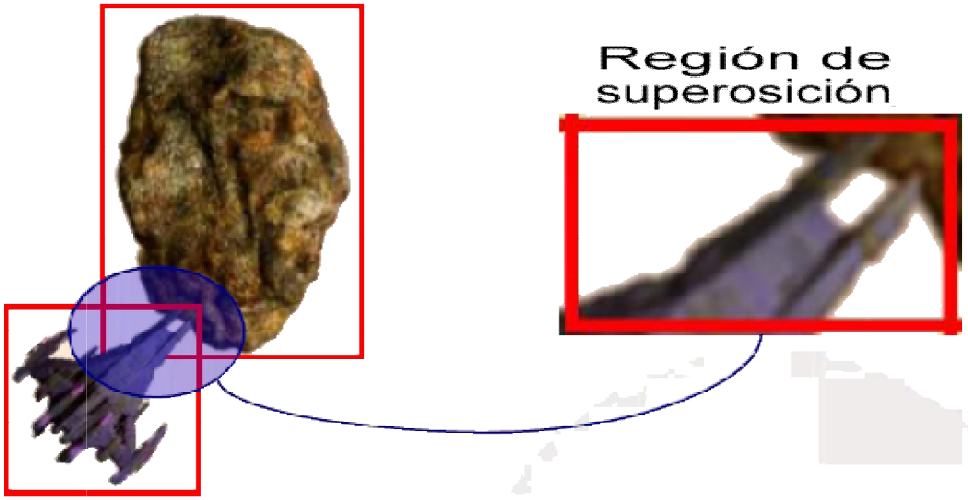
Detección de colisiones - Pixel Perfect

La técnica **pixel perfect collision** se puede implementar utilizando un mapa de colisiones, para reducir el número de píxels y así el cálculo sea menos costoso. Se trabaja con una imagen de menor resolución, o mapa de colisiones.
El mapa de colisiones tiene muchos menos píxeles, y por tanto menos comprobaciones y cálculos a realizar.
Para generar el mapa de colisiones, se pueden generar dos bitmaps (detallado y pixelado), y se cargan ambos en el sprite, se usa uno para mostrar por pantalla, y otro para calcular las colisiones.

Técnicas y algoritmos para VJ2D

Detección de colisiones

Pixel Perfect



Técnicas y algoritmos para VJ2D

Detección de colisiones

Los métodos de detección de colisiones pueden ser clasificados en:

- A posteriori
- A priori

Técnicas y algoritmos para VJ2D

Detección de colisiones

A posteriori

Los elementos del juego, y el mismo entorno avanzan a pequeños intervalos de tiempo, al avanzar se chequea si hay colisiones y si ocurren se actualiza la posición para que la colisión no se visualice.

Es un algoritmo simple de implementar, ya que no actualiza de forma real la posición de los elementos que colisionan sino simplemente los reubica, o los deja en la posición anterior.

Tiene mejor performance, pero le quita realismo al juego.

Técnicas y algoritmos para VJ2D

Detección de colisiones

A priori

El algoritmo de detección esta implementado para predecir precisamente las trayectorias de los objetos cuando colisionan, las colisiones son calculadas antes de que se produzcan.

Tienen como ventaja que brindan mayor estabilidad y fidelidad a la realidad, pero se necesitan algoritmos complejos que los implementen.

Diseño
Segunda vuelta

Andrés Pastorini - Alvaro Martínez

TDIVJ2D – Tecnólogo Informático

Técnicas y algoritmos para VJ2D

Diseño

Modelado de software

Se deben modelar entidades que representen y que permitan manejar:

- AI juego
- Animaciones
- Sonidos
- Eventos
- Detección de colisiones
- Inteligencia artificial

Diseño

Modelado de software

Los elementos que aparecen en un juego son

- Pantalla
  - Barra de estado
    - Puntuación, vidas, mapas, etc.
  - Ventana de juego
    - Sprites, cursor, fondo, etc.
- Sonidos

Técnicas y algoritmos para VJ2D

Diseño

Modelado de software

Para diseñar un juego se podría utilizar una entidad principal de juego (game class) que se ocupe de:

- Manipular la interfaz con el video, y el audio
- Contenga y controle todos los sprites
- Maneje o delegue los eventos del usuario
- funcionar en base al tiempo transcurrido (main loop)
- Manejar la lógica del juego

Técnicas y algoritmos para VJ2D

Diseño

Modelado de software

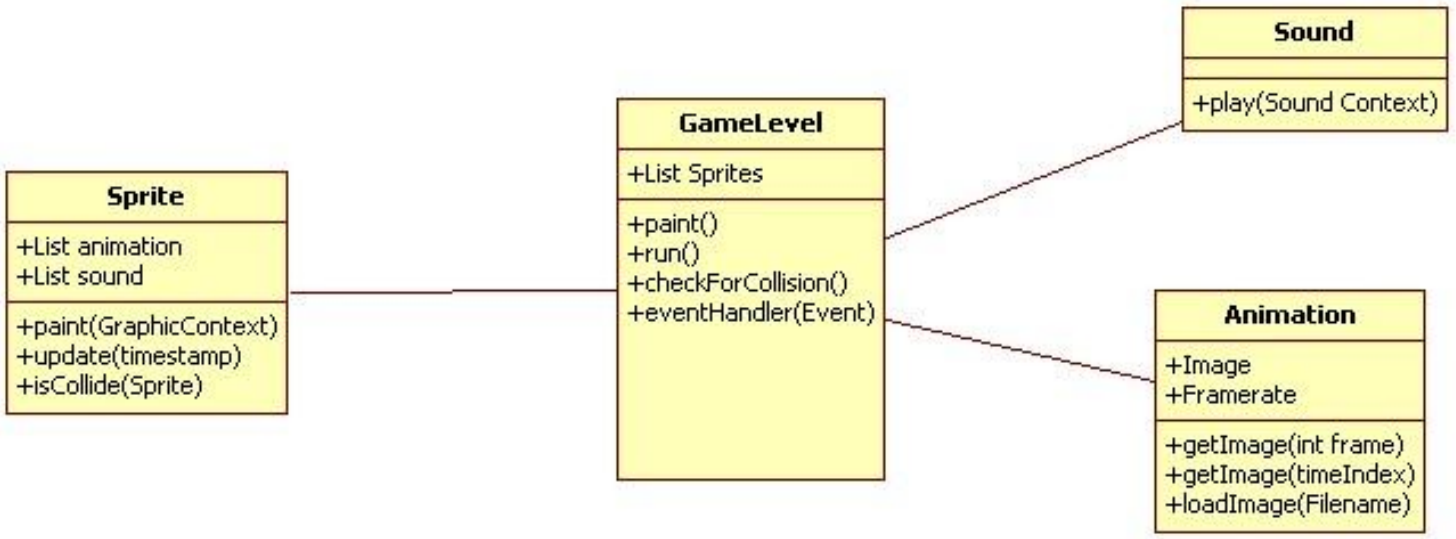
Entidad(Sprite) que represente a los diversos personajes del juego para:

- Manejar su estado
- Posición
- Dimensiones
- Renderizarse
- Ejecutar animaciones y sonidos

Técnicas y algoritmos para VJ2D

Diseño

Modelado de software



Técnicas y algoritmos para VJ2D

Diseño

Modelado de software

Es común a la hora de diseñar videojuegos, pensar distintas partes del mismo con una maquina de estados.

Técnicas y algoritmos para VJ2D

Diseño

Modelado de software

Los estados se pueden implementar de tres formas diferentes:

- **clausulas if/else o switch/case**
- **tablas**
- **design pattern state**

Técnicas y algoritmos para VJ2D

Diseño

Main loop

Es el código que organiza la ejecución de todos los componentes del juego, durante toda la ejecución del mismo.

Main loop:

```
Mientras se desea jugar
    Manejo del tiempo
    Obtener eventos
    Actualizar la posición de los sprites y el mundo
    Procesar Colisiones
    Actualizar Estados y ejecutar acciones
    Actualizar pantalla
```

Técnicas y algoritmos para VJ2D

Diseño

Main loop

```
while True:
    #-----------------------------------------------------------------#
    # Gestionar la velocidad del juego
    #-----------------------------------------------------------------#
    if pygame.time.get_ticks()-tiempo < 1000/fps:
    continue
    tiempo = pygame.time.get_ticks()
    #-----------------------------------------------------------------#
    # Bucle de eventos: Mirar si se quiere terminar el juego
    #-----------------------------------------------------------------#
    for evento in pygame.event.get():
    if evento.type == KEYDOWN:
    if evento.key == K_ESCAPE:
       pygame.quit()
       sys.exit()
    # Mira si el jugador apreto una tecla
    teclasPulsadas = pygame.key.get_pressed()
    procesarInput(teclasPulsadas)
    actualizarEstadoDeJugo()
    dibujarPantalla()
```

Técnicas y algoritmos para VJ2D

Diseño

Main loop

```
Def procesarInput(teclasPulsadas):

if teclasPulsadas[K_a]:
   #accion1
if teclasPulsadas[K_q]:
   #accion2
```

Patrones de Diseño

Andrés Pastorini - Alvaro Martínez

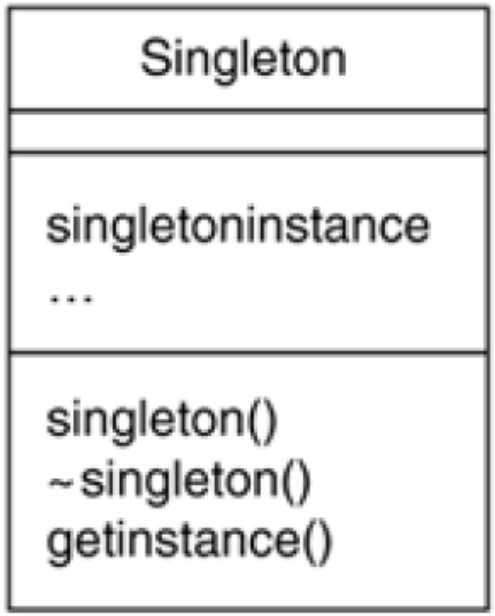
TDIVJ2D – Tecnólogo Informático

Patrones de diseño para videojuegos

Singleton

Muchas veces es necesario tener objetos globales que puedan ser visibles desde cualquier otro objeto.
Algunos posibles ejemplos:

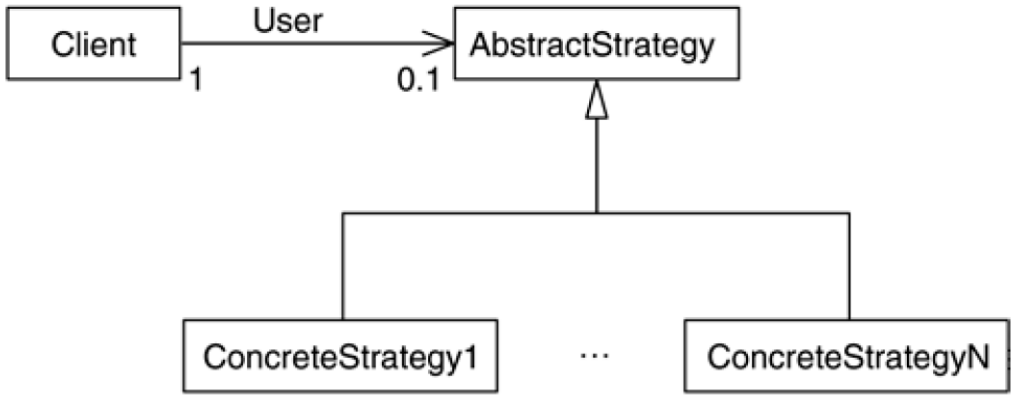
- Cargador de imágenes
- Manejador de texturas
- Controlador de joystick
- Clase jugador



Patrones de diseño para videojuegos

Strategy

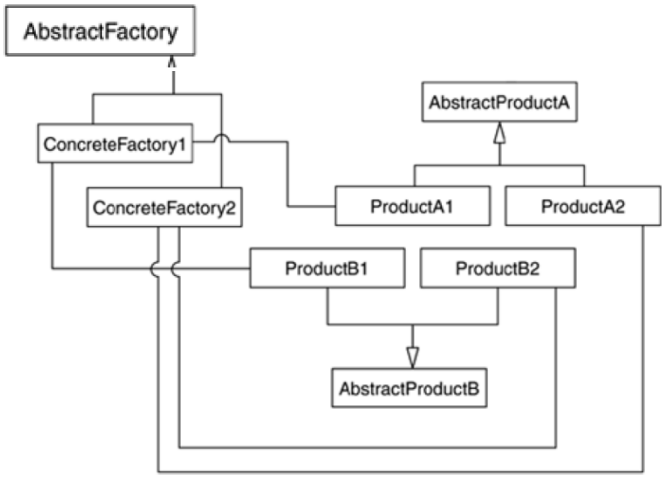
Cuando es necesario crear un elemento(personaje, enemigo,etc.) del juego cuyo comportamiento pueda ser cambiado dinámicamente.



Patrones de diseño para videojuegos

Factory

La mayoría de las aplicaciones necesitan crear y eliminar objetos continuamente, por ejemplo un juego podría estar creando y eliminado enemigos continuamente. Este patrón centraliza la creación y destrucción de objetos.



Técnicas y algoritmos para VJ2D

Patrones de diseño para videojuegos

Composite

Puede ser necesario mantener colecciones de datos heterogéneos juntos por distintas razones. Por ejemplo un nivel del juego puede tener subniveles, enemigos, objetos, etc.

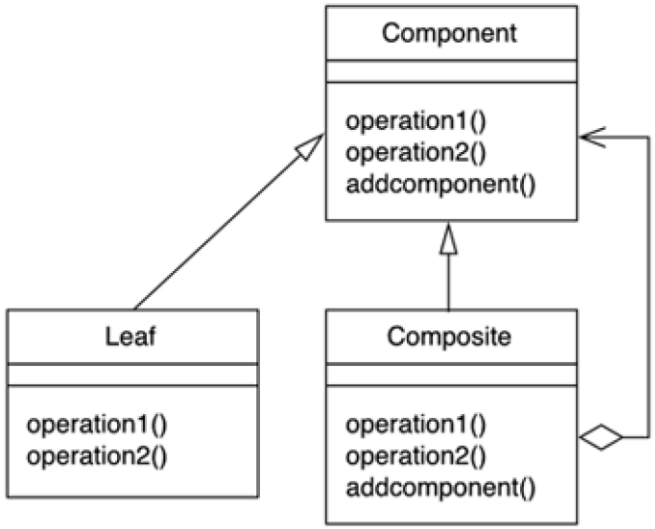
La estructura de datos entera puede ser mejor descrita de una forma jerárquica como parte-todo, donde cada elemento puede ser primitivo o compuesto

Técnicas y algoritmos para VJ2D

Patrones de diseño para videojuegos

Composite

Este patrón nos permite trabajar tanto con objetos simples, como con objetos compuestos utilizando una interfaz única.



Técnicas y algoritmos para VJ2D

Patrones de diseño para videojuegos

Flyweight

Este patrón es útil cuando se necesitan mantener grandes colecciones de objetos que son idénticos entre si salvo algunos parámetros.

Por ejemplo en un juego de estrategia en tiempo real, en los distintos niveles los soldados son exactamente iguales salvo que tienen algunas variantes. Quizás varíe su la ubicación geográfica y las armas que utilizan en los distintos nivel por ejemplo, después su comportamiento, las imágenes asociadas, las armas son los mismos.

Técnicas y algoritmos para VJ2D

Patrones de diseño para videojuegos

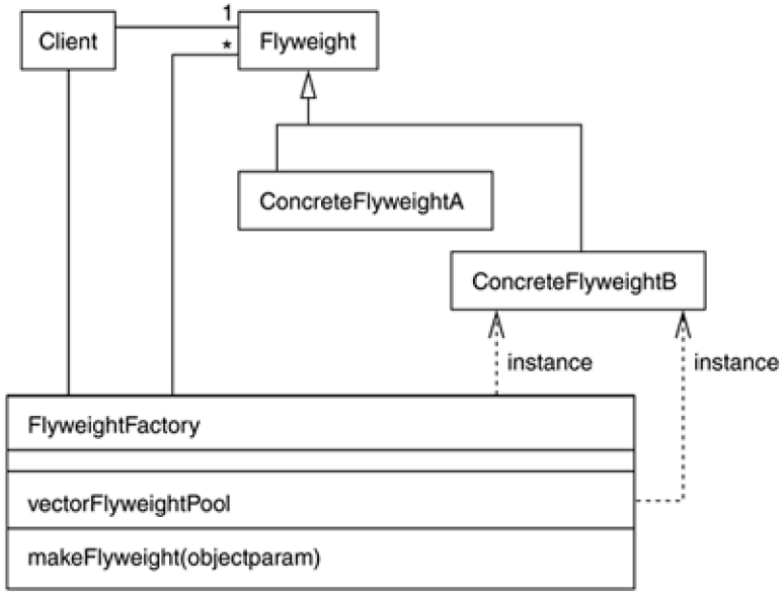
Flyweight

El patrón sugiere dividir el objeto en dos clases separadas, primero se crea el objeto ***flyweigth*** que es el objeto compartido por todas las instancias ya que contiene la funcionalidad genérica. Esta clase es manejada a través de ***Flyweigth factory.***

Patrones de diseño para videojuegos

Flyweight

Luego los objetos concretos (**ConcreteFlyWeigthX**) serán los que tengan la funcionalidad y estados específicos para cada nivel.

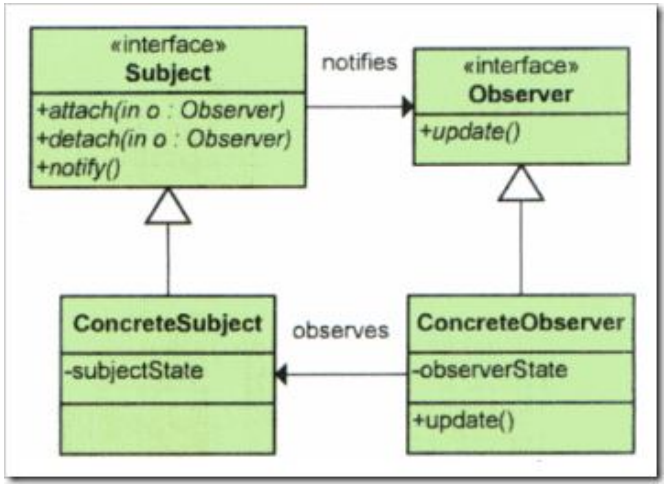


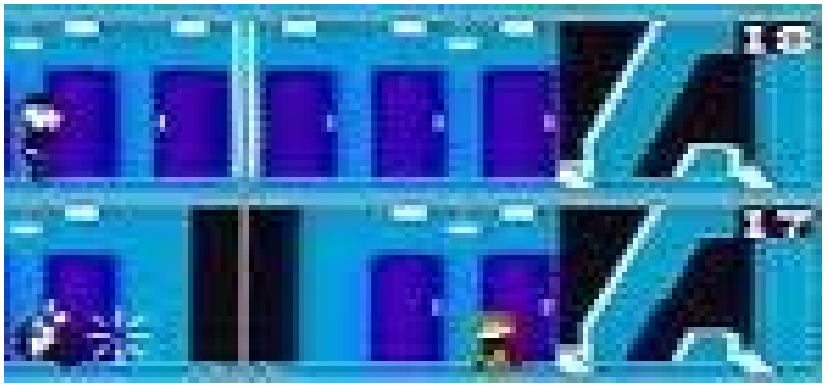
Técnicas y algoritmos para VJ2D

Patrones de diseño para videojuegos

Observer

Las distintas entidades del juego en vez de iterar para consultar datos del mundo y de las otras entidades, podrían estar registradas como observadores en los elementos cuyo comportamiento las podría afectar.





Técnicas y algoritmos para VJ2D

Core Techniques and Algorithms in Game (Cap. 4 y Cap. 11)Programming [Paperback]-Daniel Sanchez-Crespo Dalmau

http://juank.black-byte.com/xna-colisiones-2d/

http://geeks.ms/blogs/jbosch/archive/2009/08/08/xna-pixel-perfect-collision-con-xna-en-base-a-un-mapa-de-colisiones-2d.aspx

http://www.innerjuegos.com/noticias/13817-triple-buffering-que-es-y-para-que-sirve

http://www.gamedev.net/page/resources/_/technical/game-programming/introduction-to-isometric-engines-r744

Técnicas y algoritmos para VJ2D

http://www.gamedev.net/page/resources/_/technical/game-programming/isometric-n-hexagonal-maps-part-i-r747

http://wikis.uca.es/wikijuegos/w/index.php?title=Temario/Los_Sprites_y_los_Personajes

http://www.efn.uncor.edu/escuelas/computacion/files/sdl.pdf

http://www.l2radamanthys.com.ar/sitios/Curso%20PyGame/capitulo1.html

http://www.coranac.com/tonc/text/regbg.htm

http://wiki.sheep.art.pl/Tiled%20Map%20in%20PyGame

Técnicas y algoritmos para VJ2D

http://www.mechanicalcat.net/richard/log/Python/PyGame_sample__drawing_a_map__and_moving_around_it

http://aventurapygame.blogspot.com/2011/09/gestion-de-imagenes-tiles.html


SCORE 0
TIME 0:06
RINGS 10
SONIC
× 2