Pseudocódigo de PayasoLanderANN by Omarsaurio (ojorcio@gmail.com) 2019

Sobre el Software:

Inspirado en el clásico Space-Lander que se trata de un videojuego casual con comandos simples: cambiar el giro del objeto y propulsarlo; un objetivo concreto: aterrizar a salvo en una plataforma o área; y unas métricas como lo son el tiempo de juego.

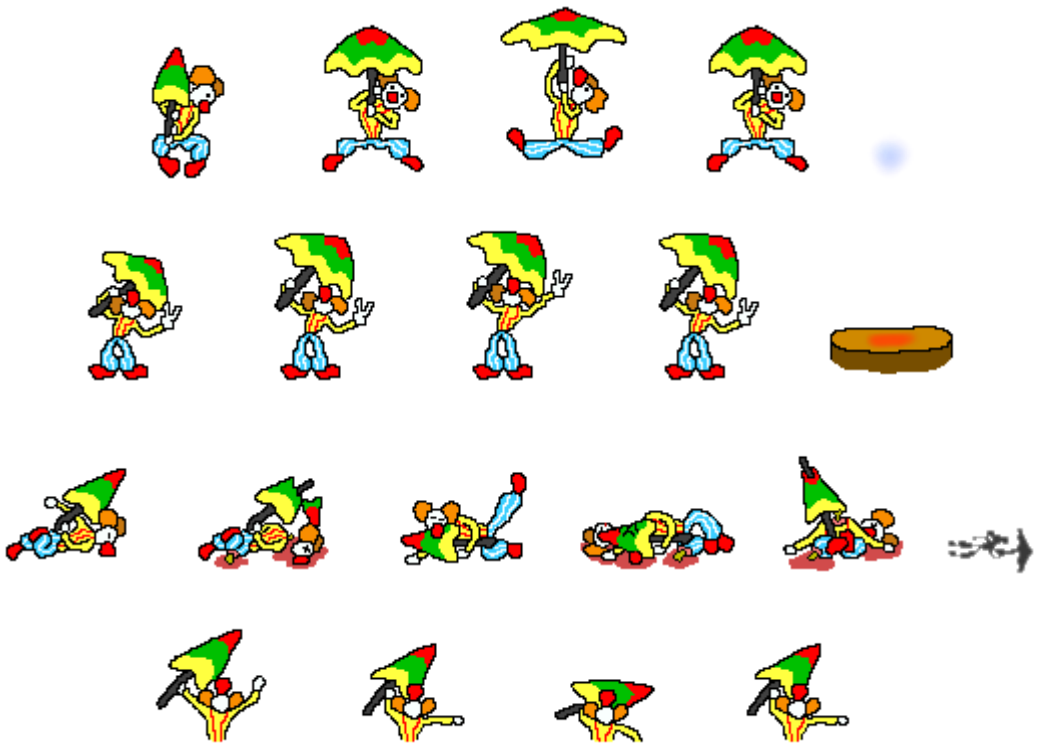
En este caso se ha cambiado la estética por la de un payaso con sombrilla que trata de aterrizar en una pequeña plataforma en el agua, puede perder si es empujado fuera de los límites de cámara, si cae al agua, si se golpea contra la plataforma o cae en un ángulo inadecuado; también debe lidiar con el consumo de energía que lo puede dejar incapaz de impulsarse.

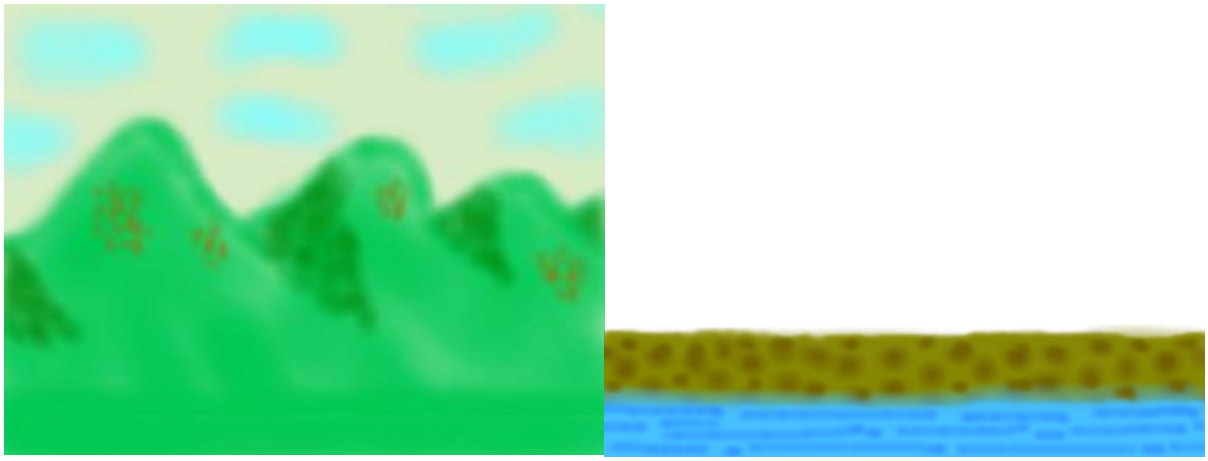
Se pueden tomar los datos en forma de archivo de texto, para entrenar máquinas de clasificación de patrones, actualmente soporta MLP y DMNN.

En el presente documento se halla el código básico que puede ser implementado en cualquier motor o lenguaje, no incluye el procesamiento del MLP y DMNN, así como tampoco las animaciones del personaje, los momentos sonoros y la especificación de la función de guardar patrones, esta última debe usarse cada cierto tiempo o cuando halla cambios en los comandos, de su accionar depende el tener un correcto set de entrenamiento.

Arte Gráfico:

No es mi especialidad, así que cree con el mouse unas imágenes animadas de baja resolución (volando, partícula viento, aterrizado, plataforma, muerto, perdido en el fondo, ahogándose, fondo y suelo acuático):





Arte Sonoro:

Se proponen los siguientes sonidos básicos, fácilmente puede notar en el pseudocódigo dónde irían:
- música de fondo circense.
- silbido al salir de los límites.
- estrellarse al caer en el fondo luego de salir de los límites.
- chapuzón al caer al agua.
- golpe y grito al caer contra la plataforma.
- quejido al resbalarse en la plataforma.
- risa y sonido de triunfo.
- quejido al quedarse sin energía.
- tenue soplido del viento al fondo.
- efecto abanico al batir la sombrilla.

Constantes:
derecha, arriba, izquierda (código): teclas (key) para ejercer comandos de giro y propulsión.
m_ve (num): velocidad de descenso de la energía al propulsarse.
m_nrg (num): cantidad máxima de energía que se posee al inicio.
m_rv (num): tiempo para calcular intervalos de cambio de la aceleración del viento.
m_av (num): aceleración del viento máxima.
m_vv (num): velocidad del viento máxima.
m_giro (num): aceleración aplicada al giro cuando se da el comando.
m_fg (num): fricción con la que la velocidad de giro disminuirá hasta detener el cambio.
m_vg (num): velocidad máxima de giro.
m_pg (num): ángulo máximo de giro que puede poseer el payaso.
m_propulsion (num): intensidad de la aceleración con que se puede impulsar el payaso.
m_gravedad (num): aceleración de la gravedad.
m_fv (num): fricción de la velocidad, oposición al movimiento, independiente del viento.
m_vxy (num): máxima velocidad que puede alcanzar el payaso.
m_px (num): máxima posición horizontal, es un poco mayor al ancho de la mitad de la ventana.
m_py (num): máxima posición vertical o altura, es un poco mayor a la altura de la ventana.
ancho, alto (num): tamaño de la ventana del software, referido al objeto ventana.
margen_inferior (num): pequeña distancia entre la plataforma y el borde inferior de la ventana.
m_radio (num): tamaño de la plataforma de aterrizaje, en total sería 2 * m_radio.
m_cerca (num): pequeña distancia entre el payaso y la plataforma para hacer contacto.

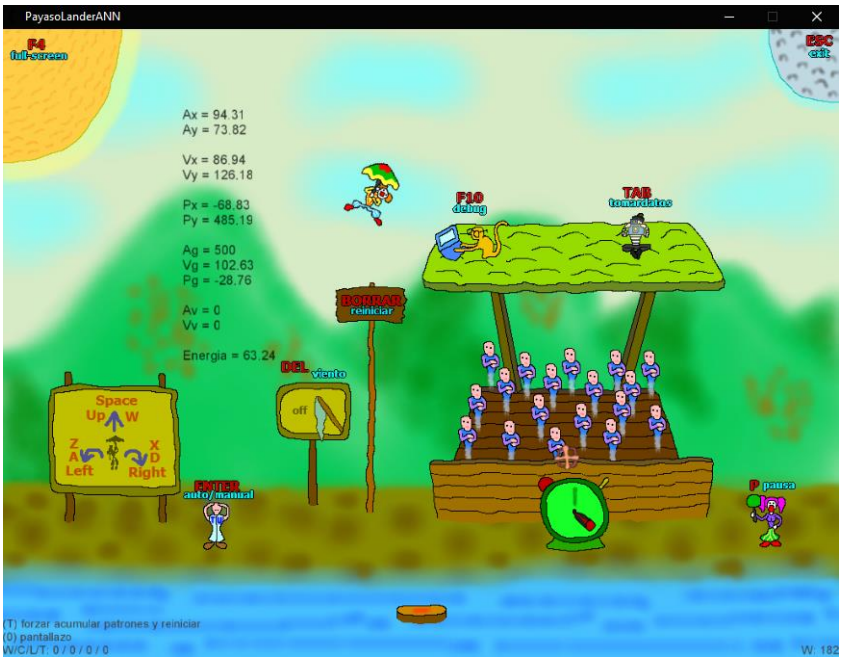
m_golpevy (num): velocidad de caída a partir de la cual el aterrizaje es muy brusco.
m_golpevx (num): velocidad en eje horizontal por debajo de la cual se puede aterrizar.
m_caeg (num): ángulo a partir del cual el aterrizaje es inadecuado.

Variables:
manual (si, no): verdadero si el payaso será manejado por el usuario.
mlp (si, no): verdadero si el payaso será controlado por una red perceptrón multicapa.
dmnn (si, no): verdadero si el payaso será controlado por una red morfológica de dendritas.
viento (si, no): verdadero si habrá influencia del viento en el juego.
tomardatos (si, no): verdadero si se desea guardar patrones para entrenar sistemas.
cp (0, 1): propulsión del payaso (no propulsar, propulsar)
cg (-1, 0, 1): dirección de giro del payaso (derecha, no girar, izquierda)
nrg (num): energía del payaso, se agota mientras esté propulsando.
reloj_v (num): contador para cambiar la aceleración del viento en intervalos al azar.
av (num): aceleración del viento, su intensidad cambia al azar.
vv (num): velocidad del viento, este solo tiene componente horizontal.
ag (num): aceleración angular, depende del comando de entrada para modificar el giro.
vg (num): velocidad angular para cambiar el ángulo del payaso.
pg (num): ángulo al que se encuentra el payaso, 0 está totalmente derecho.
ax, ay (num): aceleración para movimiento, depende del comando de propulsión y la gravedad.
vx, vy (num): velocidad del payaso, aquí se agrega además la influencia del viento.
px, py (num): posición del payaso respecto a la plataforma de aterrizaje, eje Y invertido.
x, y (num): posición del payaso respecto a la pantalla del software (absoluta).

Muestreo de Patrones:

La idea de la última función es que tome los datos cada vez que haya un cambio en cp o cg, así como también cada cierto tiempo (ej: 500ms), tiempo reseteable si hubo el cambio; también hay zonas de la pantalla donde podría agregarse una componente de azar, para o sobrecargar esas zonas de datos.

Finalmente, si no desea tomar datos o simular el juego automáticamente, evite las líneas 3, 6 a 9 y 49 del código.



```
// inicio ciclo
// manejo del payaso
si manual
       {si key.arriba     cp = 1     sino                       cp = 0}
       {si key.derecha    cg = -1    sino si key.izquierda      cg = 1    sino    cg = 0}
sino si mlp
       cp, cg = MLP(px, py, vx, vy, pg, vv)
sino si dmnn
       cp, cg = DMNN(px, py, vx, vy, pg, vv)
// limitar energía
si cp = 1
       {si nrg = 0    cp = 0    sino    nrg = limitar(nrg - m_ve * dt, 0, m_nrg)}
// calcular viento
si viento
       reloj_v -= dt
       si reloj_v <= 0
              reloj_v = m_rv + random(m_rv)
              av = -m_av + random(2 * m_av)
       vv = limitar(vv + av * dt, -m_vv, m_vv)
sino si vv !=0    vv = 0
// físicas giro
ag = cg * m_giro
vg = limitar(vg * m_fg + ag * dt, -m_vg, m_vg)
pg = limitar(pg + vg * dt, -m_pg, m_pg)
// físicas movimiento
ax = -sen(pg) * cp * m_propulsion
ay = m_gravedad - cos(pg) * cp * m_propulsion
vx = limitar(vx * m_fv + vv + ax * dt, -m_vxy, m_vxy)
vy = limitar(vy * m_fv + ay * dt, -m_vxy, m_vxy)
px = limitar(px + vx * dt, -m_px, m_px)
py = limitar(py + vy * dt, -m_py, 0)
// coordenadas respecto a pantalla (0,0 esquina superior izquierda)
x = ventana.ancho / 2 + px
y = ventana.alto - margen_inferior + py
// verificación victoria o muerte
si px = m_px | px = -m_px | py = -m_py    fuera_limites()
sino si abs(px) > m_radio                 {si py = 0    nadando()}
sino si abs(py) <= m_cerca
       si vy >= m_golpevy
              si pg < -m_caeg        golpeado_derecha()
              sino si pg > m_caeg    golpeado_izquierda()
              sino                   golpeado()
       sino si abs(vx) <= m_golpevx
              si pg < -m_caeg        resbalado_derecha()
              sino si pg > m_caeg    resbalado_izquierda()
              sino                   gano_juego()
       sino    vy = -m_golpevy
// guardar patrones
si tomardatos    guardar_patrones(px, py, vx, vy, pg, vv, cp, cg)
//fin ciclo
```