RUN OR FALL

Donè Francesco - [5A 2015-16] - ITIS V.Volterra San Donà di Piave

Indice

Cos’è Run or Fall?

Run or Fall è un videogioco a piattaforme (platformer) ovvero dove la meccanica del gioco implica l’attraversamento di livelli costituiti da piattaforme a volte disposti su più piani, sviluppato graficamente in stile 2.5D, quindi utilizzando tecniche di grafica bidimensionale per creare l'illusione della tridimensionalità. La terza dimensione avanza dallo schermo verso lo sfondo, per creare il senso della profondità e dello spostamento su più piani. La peculiarità consiste nell’utilizzare il proprio smartphone come controller di gioco attraverso l’utilizzo di una connessione remota stando collegati nella stessa rete wifi. Lo scopo è piuttosto banale: il giocatore deve correre il più velocemente possibile verso la fine del percorso, evitando di cadere o di rimanere ingannato da alcune trappole. Come da titolo, Run or Fall significa letteralmente “Corri o Cadi” perciò per vincere è necessario dimostrare molta abilità nel superare tutti gli ostacoli presenti; Si può giocare da uno a due giocatori ed essendo multipiattaforma, è compatibile con la maggior parte dei dispositivi quali Android, Linux, iOS, Mac OS X, Windows Phone e Windows 7/8.1/10.

Perché è stato sviluppato?

Oggi il mercato dell'IoT (Internet of Things) è in crescita e come ha spiegato il professore di Sociologia dei processi culturali dell'università di Catania Davide Bennato: "L'espressione Internet of Things indica una famiglia di tecnologie il cui scopo è rendere qualunque tipo di oggetto, anche senza una vocazione digitale, un dispositivo collegato ad internet, in grado di godere di tutte le caratteristiche che hanno gli oggetti nati per utilizzare la rete. Attualmente le proprietà degli oggetti connessi sono essenzialmente due: *il monitoraggio e il controllo*.", perciò Run or Fall rappresenta in che modo si possono eliminare i classici Joystick o Joypad, sostituendoli con lo smartphone, in quanto grazie ad internet, esso è concettualmente in grado di controllare e monitorare oggetti (in questo caso virtuali). Come si può notare, il controllo remoto da parte dei dispositivi mobili è molto richiesto, specialmente se utilizzato con piattaforme quali *Node.js* e *Socket.io*, si parla che la dimensione economica globale del mercato dell'IoT è stata valutata in 157 miliardi di dollari; mentre nel 2021, secondo l'agenzia Research and Markets, salirà a 661 miliardi. Il relativo tasso annuo di crescita composto (CAGR) sarà di un mostruoso 33,3 per cento.

Come funziona?

Il principio di funzionamento dell'IoT è piuttosto banale: il dispositivo che comanda gli oggetti instaura una piccola connessione di tipo Client-Server, in seguito grazie all'utilizzo dei *socket*, il dispositivo è in grado di inviare e ricevere dati da e verso l'oggetto in questione. Un socket quindi, nel mondo dell' informatica, indica un'astrazione software progettata per poter utilizzare delle API (*Application Programming Interface*) standard e condivise per la trasmissione e la ricezione di dati attraverso una rete. È il punto in cui il codice applicativo di

un processo accede al canale di comunicazione per mezzo di una porta, ottenendo una comunicazione tra processi che lavorano su due macchine fisicamente separate. Dal punto di vista di un programmatore un socket è un particolare oggetto sul quale leggere e scrivere i dati da trasmettere o ricevere. Dopo aver inoltrato i dati all'oggetto da monitorare, quest'ultimo analizza i pacchetti ed esegue le operazioni in base a come è stato progettato. Gli ambiti di applicazione dell'Internet of Things sono praticamente illimitati: dalla domotica, all'agricolutra, dalle automobili alle Smart City.

Con cosa è stato sviluppato?

Il gioco è stato sviluppato solo ed esclusivamente con *Game Maker Studio*. Game Maker è un ambiente di sviluppo integrato (*integrated development environment*) per lo sviluppo di videogiochi, originariamente creato dal professor Mark Overmars e successivamente sviluppato da YoYo Games. Game Maker ("creatore di giochi" in inglese) mette a disposizione due metodi

di programmazione: a icone e a codice. Il primo è rivolto ai principianti dove grazie ad icone da trascinare con il mouse è possibile creare giochi anche senza conoscere nessun linguaggio di programmazione. Il secondo ci mette a disposizione il *Game Maker Language* (GML), un linguaggio di programmazione con la sintassi basata sull'unione dei linguaggi Delphi, Java, Pascal, C e C++. È possibile estendere le funzioni del programma, utilizzando i file *Dynamic-link library* (DLL): questi file devono essere compilati in C++, Delphi, Pascal, ASM o altri linguaggi di programmazione. È inoltre possibile creare delle *GEX*, vale a dire delle librerie scritte in GML da usare solo ed esclusivamente in Game Maker. La potenza di tale software permette con un solo codice sorgente, l'esportazione dell'applicazione per le seguenti piattaforme: Android, iOS, Windows Phone, Tizen, Windows, Linux, Mac OS X, Windows 8 App, HTML5, PSVita, PS3, PS4, XBOX One e Amazon Fire.

Specifiche tecniche

Per programmare con Game Maker Studio sono necessari 512 MB di memoria RAM (Random Access Memory) e 128 MB di memoria GPU (Graphics Processing Unit), mentre per quanto concerne l'applicativo compilato sono necessari 1 GB di memoria RAM e 256 MB di memoria GPU. GMS permette inoltre la compilazione del gioco per i monitor dove la risoluzione è pari a 4096x2304 (ovvero 4k). Un difetto che rende il gioco "lento" è il fatto che la gestione delle risorse e della memoria RAM viene gestita automaticamente: questo semplifica tantissimo lo sviluppo delle app ma le rende anche meno efficienti in quanto a prestazioni. Esistono due tipi di applicazioni per quanto riguarda Run or Fall: esse sono il *Player* (ovvero la versione desktop) ed il *Controller* (ovvero la versione mobile – "il dispositivo da cui si comandano gli oggetti"): nella versione desktop viene instaurato un *server di tipo TCP* in attesa di connessioni (*listen*) nella porta 6510, quindi

l'applicativo in questione risulta essere composto da un lato server e da un lato client e la creazione del server TCP viene eseguita instaurando un oggetto (*ObjServer*):

```
var Type, Port, MaxClients;
Type = network_socket_tcp;
Port = 6510;
MaxClients = 2;
Server = network_create_server(Type, Port, MaxClients);
var Size, Type, Alignment;
Size = 1024;
Type = buffer_fixed;
Alignment = 1;
Buffer = buffer_create(Size, Type, Alignment);
SocketList = ds_list_create();

var type_event = ds_map_find_value(async_load, "type");
switch(type_event) {
  case network_type_connect:
    var socket = ds_map_find_value(async_load, "socket");
    ds_list_add(SocketList, socket);
  break;
  case network_type_disconnect:
    var socket = ds_map_find_value(async_load, "socket");
    var findsocket = ds_list_find_index( SocketList,
socket);
    if (findsocket >= 0) {
      ds_list_delete(SocketList, findsocket);
    }
  break;
  case network_type_data:
    var buffer = ds_map_find_value(async_load, "buffer");
    var socket = ds_map_find_value(async_load, "id");
    buffer_seek(buffer, buffer_seek_start, 0);
    ReceivedPacket(buffer, socket);
  break;
}
```

In seguito nella versione mobile (Controller) viene creato un *server di tipo UDP*, grazie al quale si inoltrano pacchetti in broadcast all'indirizzo 255.255.255.255: quando si riceve una risposta, basta analizzare il pacchetto per vedere *l'indirizzo ip locale* del mittente, ovvero l'indirizzo ip locale del server TCP creato dal Player. Quando si è in possesso dell'ip del server TCP è

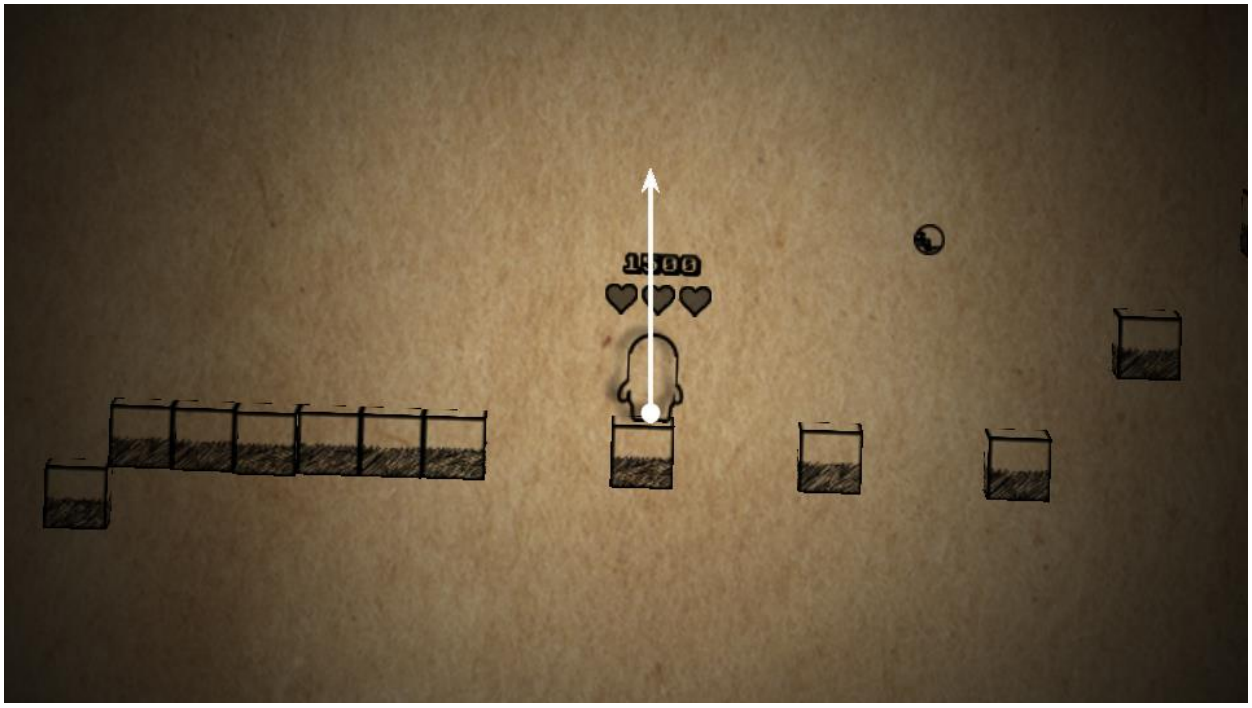
possibile effettuare una connessione attraverso i socket e si può cominciare a monitorare il proprio giocatore inoltrando informazioni attraverso l'utilizzo dei buffer. Tutto ciò viene eseguito grazie all'oggetto dedicato rinominato *ObjClient*:

```
var Type, IPAddress, Port;
Type = network_socket_tcp;
IPAddress = "127.0.0.1";
Port = 6510;
Socket = network_create_socket(Type);
isConnected = network_connect(Socket, IPAddress, Port);
var Size, Type, Alignment;
Size = 1024;
Type = buffer_fixed;
Alignment = 1;
Buffer = buffer_create(Size, Type, Alignment);
var type_event = ds_map_find_value(async_load, "type");
switch(type_event) {
  case network_type_data:
    var buffer = ds_map_find_value(async_load, "buffer");
    buffer_seek(buffer, buffer_seek_start, 0);
    ReceivedPacket(buffer);
  break;
}
```

La dimensione dei buffer utilizzati è di 1024 byte in quanto una dimensione inferiore risulterebbe propensa ad effettuare il fenomeno chiamato *bottleneck*. In ingegneria del software, il collo di bottiglia o bottleneck è un fenomeno che si verifica quando le prestazioni di un sistema o le sue capacità sono fortemente vincolate da un singolo componente. Il componente viene spesso chiamato anch'esso collo di bottiglia o punto del collo di bottiglia. Il termine è una metafora del collo di bottiglia reale, che limita il flusso d'uscita dell'acqua.

Per quanto concere la fisica applicata al gioco, è stata "virtualizzata" la forza di gravità attraverso l'utilizzo delle equazioni differenziali appartenenti alla famiglia a due parametri: per convenzione il vettore di riferimento è stato preso con direzione positiva, crescente verso l'alto, inoltre l'origine combacia con lo stato del corpo nell'istante *x(0)*. Di conseguenza

il riferimento di spostamento nel piano cartesiano sarà il seguente:



Nell'istante di partenza, il corpo si trova nel punto 0 ed è spinto verso l'alto con una forza pari a 3.75 m/s. La forza contraria, ovvero quella che preme verso il basso è la forza di gravità che per semplificare, è stata impostata a 10 m/s².

$$x(0) = 0$$

$$\dot{x}(0) = 3.75m/s$$

$$F = ma \qquad -ma = -mg = -m\ddot{x}$$

$$\begin{cases} \ddot{x} = -g = -10 \\ x(0) = 0 \\ \dot{x}(0) = 3.75 \end{cases}$$

Essendo *x(t)* la funzione rappresentante la posizione del corpo nell'istante *t*, la funzione che indica la velocità è la derivata prima della funzione sopraccitata, mentre la derivata seconda indica l'accelerazione che in questo caso vale -10 m/s² in quanto la sua direzione è opposta rispetto al vettore di riferimento. Integrando più volte, si ricavano le funzioni *x* e *z* che in seguito sono state utilizzate per calcolare l'altezza massima che il corpo raggiunge,

il tempo necessario per raggiungere il terreno e la velocità di impatto con il sottosuolo.

$$z = \dot{x} \qquad \dot{z} = \ddot{x} \qquad \dot{z} = -10$$

$$\frac{dz}{dt} = -10 \qquad \int dz = \int dt - 10 \qquad z = -10t + c1$$

$$\frac{dx}{dt} = -10t + c1 \qquad \int dx = \int (-10t + c1)dt \qquad x = -\frac{10}{2}t^2 + c1t + c2$$

Grazie alle funzioni *x* e *z* è possibile calcolare il valore sei parametri *c1* e *c2* ovvero la velocità del corpo nell'istante *x(0)* e la posizione dello stesso nel medesimo istante.

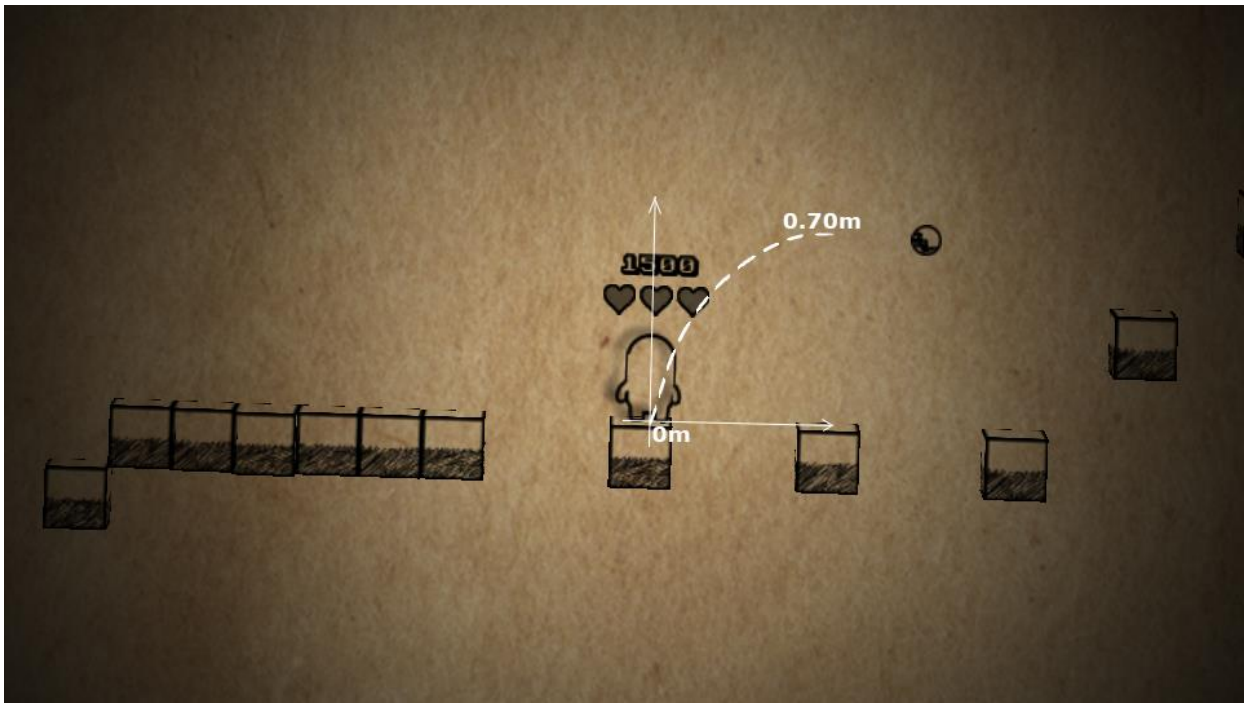
$$\begin{cases} z = -10t + c1 \\ x = -5t^2 + c1t + c2 \end{cases} \qquad \begin{cases} 3.75 = 0 + c1 \\ 0 = 0 + 0 + c2 \end{cases} \qquad \begin{cases} 3.75 = c1 \\ 0 = c2 \end{cases}$$

$$x(t) = -5t^2 + c1t + c2 \qquad x(t) = -5t^2 + 3.75t + 0$$

Di conseguenza per calcolare l'altezza massima che il corpo raggiunge, basta trovare il tempo necessario al corpo per raggiungere una velocità pari a *0m/s* (ovvero nel momento in cui esso comincia a scendere verso il basso). In seguito grazie alla funzione *x(t)* si trova la posizione di esso nell'istante trovato in precedenza:

$$\begin{aligned} &Altezza\,Massima \\ &0 = -10t + 3.75 \\ &10t = 3.75 \\ &t = 0.37 \\ &x(0.37) = -0.70 + 1.40 + 0 \\ &x(0.37) = 0.70m \end{aligned}$$

Risulta quindi che l'altezza massima raggiunta dal giocatore nella fase di salto è pari a ***0.70m***:



Per quanto riguarda il tempo necessario per raggiungere il terreno dal momento che il corpo salta, è sufficiente risolvere l'equazione descritta dalla funzione *x(t)* nell'istante in cui la funzione stessa vale 0, ovvero nell'istante in cui il corpo raggiunge il terreno. Dopo aver trovato le soluzioni, è necessario escluderne una, ovvero quella in cui il tempo assume un valore *non accettabile*:

$$
\begin{aligned}
&Tempo\,Totale\\
&0 = 5t^2 + c1t + c2\\
&0 = 5t^2 + 3.75t + 0\\
&t_1 = 0 Non Accettabile\\
&t_2 = 0.75s
\end{aligned}
$$

Risulta quindi evidente che il corpo, rimane sopraelevato rispetto al terreno per un intervallo di tempo pari a **0.75s**.

Per quanto concerne la velocità di impatto con il sottosuolo invece, è necessario applicare la funzione *z(s)* nell'istante *s=0.75* ovvero nel momento in cui il corpo raggiunge il sottosuolo:

$$
\begin{aligned}
&Velocità\,di\,Arrivo\\
&\dot{x} = -10t + 3.75\\
&\dot{x}(0.75) = -10 \cdot 0.75 + 0\\
&\dot{x}(0.75) = -7.5m/s
\end{aligned}
$$

Il risultato è pari a **-7.5m/s**, ovviamente è negativo in quanto è opposto al vettore di riferimento cartesiano.

Conclusioni

Run or Fall è un prototipo di gioco in cui vengono impiegate le tecniche di sviluppo classiche dell'Internet of Things, il quale è una frontiera esponenzialmente in sviluppo, soprattutto all'estero. Rappresenta in che modo gli smartphone possono risultare veri e propri oggetti programmabili. Con le console (quali PlayStation e XboX) c'è il limite legato ai controller, perchè mediamente ogni console ne possiede due; Run or Fall invece, utilizzando lo smartphone come controller, aggira tale problematica in quanto ognuno di noi (circa) ne possiede almeno uno. Ciò dimostra che oggi gli smartphone non sono semplici cellulari, ma veri e propri computer programmabili, che possono essere impiegati in qualsiasi settore: dallo studio, al gioco, senza costi aggiuntivi.