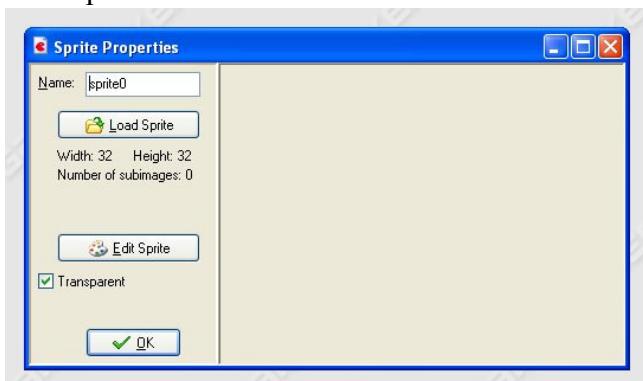


més coses a la web <http://famad.wordpress.com>

## Les imatges que formaran el joc i els seus objectes en el Mode Avançat

Si fem memòria i recapitem, tot agafant informació ja publicada per nosaltres, veurem com podem afegir imatges o **crear-les des de zero** tot **modificant-les** des de la mateixa aplicació. Ens centrarem doncs en la creació i el disseny dels propis **Sprites** accedint al

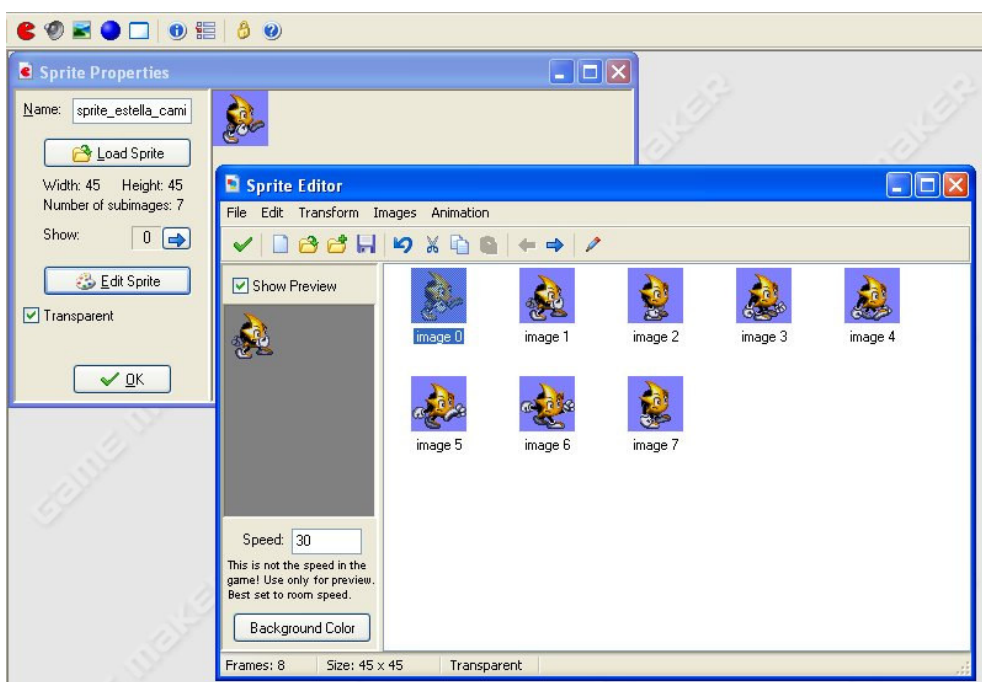


quadre de propietats que ja coneixem. Per arribar-hi haurem d'obrir la finestra del **Sprite** que vulguem modificar fent-hi doble click amb el ratolí o crear-ne un de nou des de les opcions del menú o la barra d'eines ja comentades en l'arxiu *Teoria del Videojoc 2, l'entorn propi de l'aplicació*

En **mode senzill** l'esmentat quadre ens ofereix les possibilitats de carregar una

imatge des del botó **Load Sprite** (cosa que ja hem fet i que coneixeu), editar-ne una de nova amb **Edit Sprite**, marcar el quadre **Transparent** perquè el fons de la imatge ho sigui i un **OK** que acceptarà l'opció. Per poder editar-ne el gràfic haurem de fer click en el botó **Edit Sprite** que ens durà a la finestra d'edició inicial des d'on se'ns mostrarà totes les imatges que el componen. En contra del que veiem amb els fons de pantalla

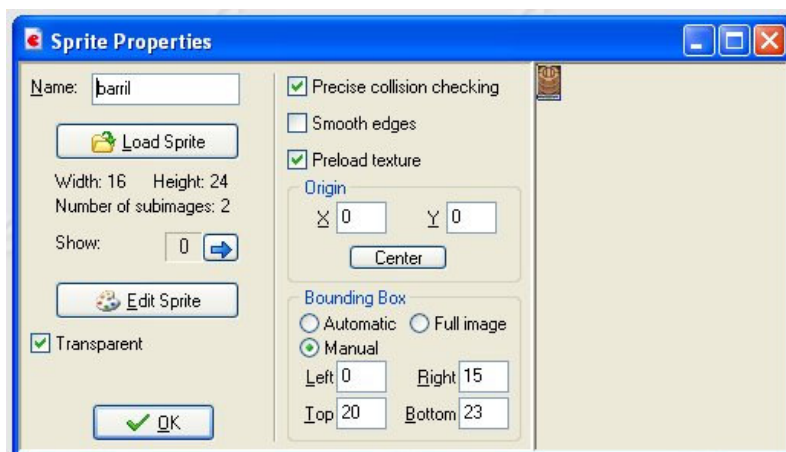
aquests **Sprites** poden ser estàtics o en moviment estant formats per una sola imatge o per un seguit d'elles que anomenarem **subimatges** i que enumerarem **des de la zero fins a la n** (l'última d'elles). Aquí, a la dreta, mostrem l'animació d'un gràfic que conté 8 moments, en cas de trobar-nos amb una imatge nova no ens apareixerà cap figura i tant sols trobarem un quadre de color verd o tapís



que haurà de contenir el nou **Sprite**. Hem de comentar que aquí, cada imatge, haurà de tenir la mateixa mida mostrant el resultat final de l'animació a la part esquerra si marquem **Show Preview** i la velocitat de reproducció al quadre **Speed** (en mode disseny i vista prèvia, doncs en el joc aquesta dependrà de la velocitat que assignem al room o nivell del joc). Cal apreciar també com el fons de la imatge és d'un color diferenciat al de la resta, doncs volem que aquest no es mostri quan juguem tot marcant la casella de transparència del quadre de propietats del mateix **Sprite**

més coses a la web <http://famad.wordpress.com>

Quan fem el **mode avançat** (per accedir-hi cal anar al menú File > Advanced Mode i l'aplicació ens mostrarà les seves opcions avançades tant en menús com en accions), ens trobem un seguit d'opcions ben diferenciades que en el mode senzill no s'ofereixen. Primerament cal observar que la part esquerra del mateix quadre es conserva quasi sense canvis, si no fos per l'opció **Show** que ens oferirà el inici de l'animació del



**Sprite**, és a dir, podem marcar a quina **subimatge** s'iniciarà el moviment. La part central del quadre s'inicia tot marcant el control sobre les col·lisions... cal que ara recordem que una imatge gràfica està dibuixada dins d'un quadre i, a vegades, aquesta no ocupa la seva totalitat deixant un espai *blanc* entre la figura i la vora

(*marge del tapís* que a la imatge es mostra en un blau o lila). Mentre en el mode senzill els **sprites** prenen una configuració estandarditzada i automatitzada, és a dir, l'objecte gràfic serà igual de gran que el quadre contenidor, que a la imatge de l'estrella mostrem puntejat en negre, i les coordenades inicials (0, 0) es situaran al cantó superior esquerre; en el mode avançat tot pot ser modificat... quan dues d'aquestes imatges topen es verifica i controla cadascun dels pixels que conformen les siluetes per veure si aquestes es sobreposen. Aquesta acció es duu a terme durant joc i, com no podria ser d'altra manera, consumeix una quantitat de memòria cabdal; que en condicions normals no serà requerida, us convidem doncs a desactivar-la sens més tot fent click a la caixa **Precise collision checking**. El sistema GM sempre l'oferirà marcada i de color verd, aquest Control precís de col·lisions, limitant d'aquesta manera la caixa al gràfic o, dit d'una altra manera, retallant la caixa per la silueta del gràfic; cosa força recomanable en jocs tipus RPG on la profunditat i la perspectiva ha de ser un valor a considerar: la imatge que us oferim forma part de l'exemple curs09.gmk, on un barril farà aleshores de límit dins del room... cal observar que s'ha modificat la caixa a l'apartat **Bounding Box** tot macant el botó **Manual** i oferint les opcions **0, 15, 20 i 23**; cosa que vol dir que l'espai de col·lisió serà de **15 pixels de llarg** per **3 d'ample**, permetent així que qualsevol figura es col·loqui per darrere del mateix sense topar-hi (bé, ho farà quan arribi al límit dels 20 pixels de la caixa que hem dissenyat a la part inferior). Altres opcions de l'apartat ens oferiran, com ja hem vist, retallar el gràfic amb **Automatic** o bé deixar la totalitat del tapís del mateix com a quadre d'acció amb **Full image**. També aquí ens permetran suavitzar les vores de la imatge tot marcant **Smooth Edges**, cosa que farà diluir més d'una filera de pixels de la silueta gràfica passant a ser transparents (rarament s'utilitza perquè és preferible tractar el gràfic anteriorment amb qualsevol programa de dibuix que deixar que l'aplicació ho defineixi de forma automàtica doncs el resultat no és gaire acceptable)

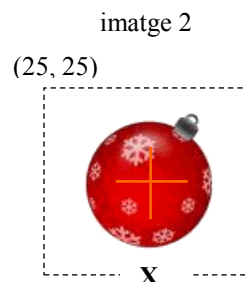
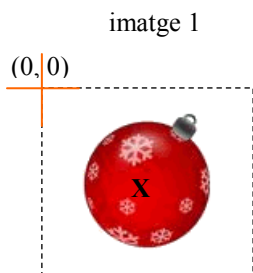
Cal deixar clar que **totes les imatges es converteixen en textures** quan l'aplicació està en ús carregant-les, d'una manera automàtica, des del executable a la memòria de vídeo (de la tarja gràfica) per poder-se utilitzar. Si marquem l'opció del quadre **Preload texture** (carregar textura prèviament) això succeeix quan l'aplicació es llença no retardant la càrrega de les imatges durant l'execució de la mateixa i fent-ho al

més coses a la web <http://famad.wordpress.com>

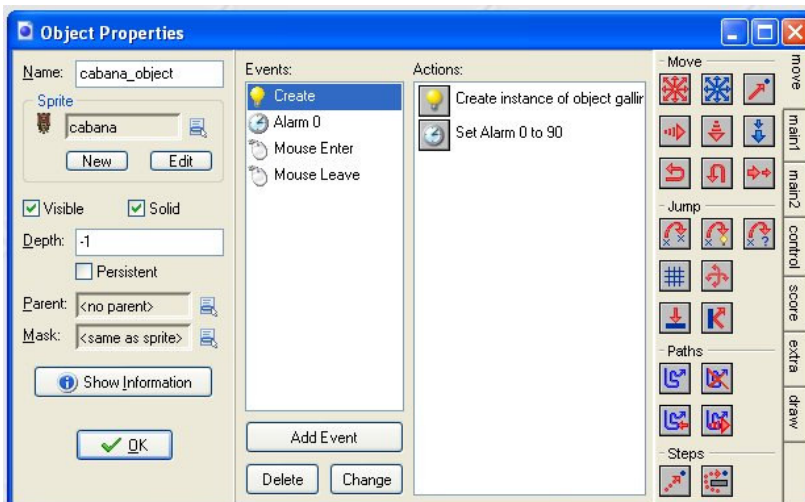
començar. Per contra, si utilitzem imatges molt grans caldrà que la desmarquem i, d'aquesta forma, es carregaran només quan el nivell ho requereixi no consumint memòria d'una vegada innecessàriament

Per finalitzar hem deixat el quadre central que ens apareix marcat com a **Origin** i ens ofereix, com sempre, les coordenades **X** i **Y**. Aquest punt serà el que marcarà la seva posició sent el referent o l'ànchora del gràfic i oferint aquella posició relativa inicial al propi objecte, és a dir, per defecte l'origen de la imatge serà el cantó superior esquerre de la mateixa i es representarà, com ja hem dit, amb (0, 0) però des d'aquí el podem modificar tot centrant-lo a la imatge o oferint-li aquella posició que ens sigui més còmode per alinear el gràfic a la malla del nivell: veiem un exemple amb el següent gràfic que representa una bola de Nadal tot suposant que el seu tapís, en línia discontinua, es totalment quadrat i mesura 50 x 50 pixels

Al crear-se un nou gràfic (com la bola vermella), s'estableix un origen que podem lliurement modificar (0, 0); doncs no és el mateix fer aparèixer una instància d'un objecte a les coordenades relatives (25, 25) de la imatge 1, que fer-ho de la 2. A la primera apareixerà l'objecte des del centre de la mateixa mentre que a la segona ho farà des de la seva part inferior (ho maquem amb una **X** en negreta, mentre que l'origen l'ensenyem amb taronja)... tot això cal tenir-ho present alhora de programar objectes o personatges que han de realitzar certes accions, en essència és el mateix però us recomanem que treballau segons l'exemple de la imatge 2 perquè ens sembla molt més aclaridor i endreçat tot marcant el botó **Center**



En aquest punt ens centrarem en la definició avançada dels objectes veient de quina manera podem modificar-los des del seu quadre de propietats (aquí us mostrem un exemple que pertany al objecte cabana del arxiu curs09.gmk). Per començar cal fer una ullada al mateix per adonar-se que les accions han augmentat considerablement apareixent opcions com ara **Depth**, **Persistent**, **Parent** i **Mask**



més coses a la web <http://famad.wordpress.com>

La **Profunditat** o **Depth**: com ja s'ha dit els gràfics que han de formar en nostre joc poden estar dissenyats per semblar en part *transparents* o *no sòlids* permetent així que qualsevol figura es col·loqui per darrere del mateix sense topar-hi, quan ho definim dins del quadre del **sprite** dotant-lo d'unes dimensions reduïdes al apartat del **Bounding Box**. Això ens servirà per poder marcar una profunditat de camp alhora de dibuixar-los



(bé, el gràfic ja haurà de tenir una certa perspectiva prèviament, com en el del exemple de l'esquerra que simula una cabana en tres dimensions -el cercle taronja en línia discontinua ens marcava la part del dibuix que tocava a terra, és a dir, el quadre del propi **Bounding Box**; si a més oferim una profunditat adequada fent que la cabana es dibuixi en un primer pla crearem la sensació de poder passar pel darrere amb el nostre personatge sempre que la seva profunditat, en mode de disseny, sigui igual o més gran), doncs els mateixos es situaran a pantalla per ordre de creació dins del nivell o seguint la profunditat que marquem dins del quadre **Depth**, fent que els objectes amb una major profunditat es dibuixin en primer pla tapant la resta (hauríem d'imaginar-nos cada nivell com una llibreta plena de pàgines transparents per omplir i cadascuna d'elles numerades amb una profunditat de major a menor). D'aquesta manera si volem que un objecte es situï per sobre dels demés haurem de programar-li una profunditat negativa, per contra, si volem que formi part del nivell o room com si es tractés del paisatge (o es trobés allunyat del primer pla) aniria bé que li oferíssim una de positiva el més gran possible... tot això és molt important alhora de programar espais que ofereixin un 3D sense ser-ho doncs es tracta d'una tècnica plana que ens dóna la possibilitat de crear espais que semblen fets amb cert volum -cal pensar que la unitat 0 de profunditat és la inicial d'un joc pla i des d'aquí sumem o retem valors per conformar l'espai del joc

Els objectes que seran **Constants** de la mà de la casella **Persistent**: ara ens centrarem en aquest concepte que convertirà en persistent a tot objecte que col·loquem en un nivell durant el joc. D'aquesta manera podrem definir una instància en el room que sempre apareixerà en tots els nivells del joc fins a ser destruït o eliminat via codi... és un concepte molt poderós i en algunes aplicacions s'utilitza per fer aparèixer el personatge principal, per exemple, durant tot el joc i en tots els seus nivells sense haver-lo de definir prèviament

Els objectes **Pares** i la seva relació amb els anomenats fills: cal que ara pensem que cada objecte pot ser l'origen o el principal d'uns altres amb la casella **Parent**. Això ho podrem utilitzar, per exemple, per evitar escriure molta programació (repetint accions) tot definint un conjunt d'objectes que es comportaran com el principal sense la necessitat d'inscriure-hi cap Event, la qual cosa que vol dir; que el principal regirà la programació de la resta i tindrà totes les accions definides que, en cadena, desenvoluparan els demés... i quan diem totes les accions ens referim a totes, si s'elimina l'objecte principal també ho farà el secundari o fill perquè en depèn. Per altra banda, resulta freqüent necessitar instàncies que realitzin algunes funcions particulars i d'altres generals, en aquest cas, també podrem utilitzar un objecte origen però cal tenir clar que les accions del fill s'executaran per sobre de les del pare si pertanyen al mateix Moment quedant la resta com a heretades

El que anomenem **Màscara** o **Mask** d'un objecte no és ni més ni menys que una representació formada per dues imatges o **sprites** que han d'actuar alhora i s'utilitza per donar un aspecte professional 3D a les nostres creacions que seran programades en dues dimensions... l'exemple que sempre es posa és el d'un tanc amb una torreta per disparar: seran dues imatges en una i funcionaran de manera independent; això és, el vehicle es mourà pel nivell lliurement i la torre ho farà, en cercles, des del seu eix dalt del tanc