Samouczek do programu Game Maker

Gra w labirynt

Napisane przez : Mark Overmars (*Copyright © 2007-2009 YoYo Games Ltd), 23.12.2009*
Tłumaczenie : Grzegorz Madejski
Korzysta z: Game Maker 8.0, Edycja Lite lub Pro, Tryb: Advanced Mode
Poziom : początkujący

Gry, w których gracz porusza się po labiryncie są bardzo popularne i bardzo łatwo robi się ja w programie GameMaker. Ten samouczek pokazuje, jak w prosty sposób stworzyć taką grę. Plusem jest to, że już po stworzeniu w grę można zagrać, a następne kroki tylko ją rozszerzają by uczynić jeszcze ciekawszą. Wszystkie potrzebne materiały do gry są w folderze `Resources`.

Pomysł na grę

Zanim zaczniemy tworzyć grę trzeba opracować pomysł na nią. To jest jeden z najważniejszych kroków w projektowaniu gry. Musi ona być ekscytująca, zaskakująca i uzależniająca. Cel gry powinien być jasny dla gracza, a interfejs i zasady – łatwe do zrozumienia i przejrzyste.

Nasza gra to gra w labirynt. Każdy pokój, który stworzymy będzie labiryntem. By z niego wyjść, gracz będzie musiał zebrać wszystkie diamenty, a następnie dojść do wyjścia. Gracz będzie musiał przy tym trochę pogłówkować – przesuwać bloki, wysadzać niektóre ściany, unikać potworów. Nowe zagadki i nowe przedmioty będą pojawiały się w kolejnych poziomach, ważne by nie ujawnić wszystkich już w pierwszym labiryncie.

W grze jest więc wiele różnych obiektów. Głównym obiektem jest postać kontrolowana przez gracza, chodząca po labiryncie. Są też ściany różnych typów (by labirynt był ciekawszy) oraz diamenty, które można zbierać. Są różne przedmioty, które można wykorzystywać do różnych rzeczy. Ważnym obiektem jest też wyjście z labiryntu. Oczywiście są też potwory, których należy unikać. Zajmiemy się tymi rzeczami po kolei.

Na początek…

Na początek, zapomnimy o diamentach. Stworzymy grę, w któ®ej gracz musi po prostu dojść do wyjścia. Są tu potrzebne trzy obiekty: gracz, ściana, wyjście. Musimy stworzyć sprajty dla nich, następnie obiekty.

Obiekty

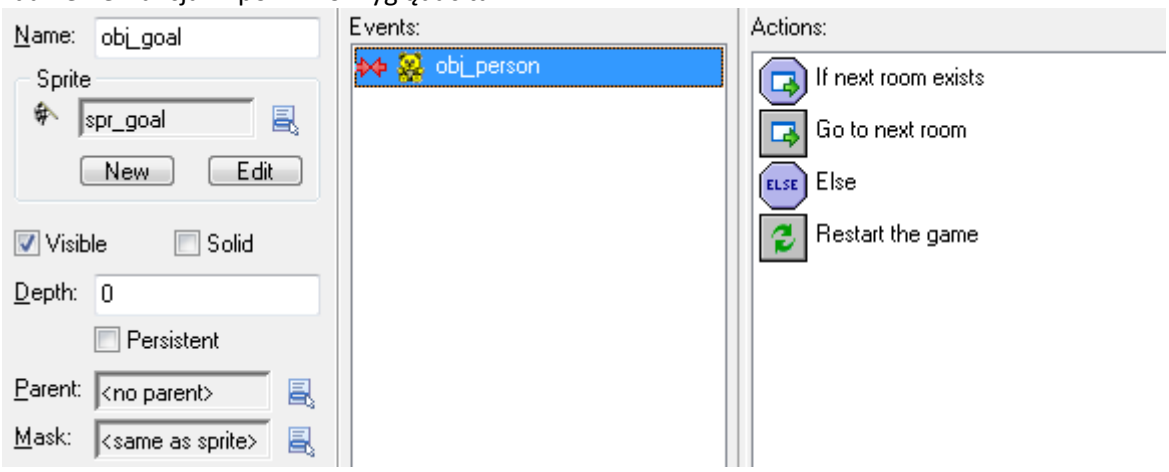
Stwórzmy zatem te 3 obiekty. Musimy oczywiście zacząć od sprajtów. Korzystając z załączonych materiałów stwórz 3 sprajty o rozmiarze 32x32 o nazwach `spr_person`, `spr_wall`, i `spr_goal`. (person = gracz, wall = ściana, goal = cel, wyjście).

Teraz stworzymy trzy obiekty na podstawie tych sprajtów.

Stwórz obiekt ściana `obj_wall` na podstawie sprajta ściany i zaznacz okienko **Solid** (stały, nieprzenikalny). Dzięki temu inne obiekty takie jak gracz nie będą mogły nigdy przechodzić przez ścianę. Poza tym ściana nic nie robi, nie musimy definiować zdarzeń i akcji.

Stwórz obiekt `obj_person` na podstawie sprajta przedstawiającego misia. Wrócimy do niego później.

Teraz stwórzmy obiekt `obj_goal` na podstawie `spr_goal`, który jest wyjściem z labiryntu (flaga finiszu symbolizuje dojście do mety, czyli do końca poziomu). Ten obiekt nie ma właściwości "solid". Gdy gracz dotknie flagi, zostanie przeniesiony do nowego pokoju. Trzeba więc dodać zdarzenie kolizji z graczem. Jest tutaj pewna pułapka. Jeśli jesteśmy w ostatnim pokoju i nie ma nowych poziomów do przejścia to pojawi się błąd. Trzeba więc zrobić test, czy istnieje następny pokój, jęsli tak to tam idziemy, jeśli nie to restartujemy grę (później ten restart gry upiększymy o dodatkowe elementy). Zdarzenie z akcjami powinno wyglądać tak:

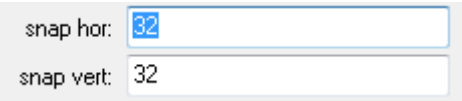


Wracamy teraz do obiektu kontrolowanego przez gracza (miś). Musimy go odpowiednio zaprogramować - miś musi reagować na klawiaturę i nie powinien wchodzić na ściany. Będzie on sterowany strzałkami (to naturalne dla każdego gracza). Są różne sposoby, by poruszać obiekt gracza. Najprostszy to przesunięcie misia o jedną komórkę w kierunku, w którym naciśnięto strzałkę na klawiaturze. Drugie podejście, które zastosujemy w naszej grze to takie, w którym gracz się porusza tak długo, jak naciśnięty jest klawisz strzałki (trzeba przytrzymać). Jest jeszcze trzecie podejście, w którym gracz porusza się cały czas a strzałki służą tylko do zmiany kierunku poruszania (podobnie jak w grach *PacMan* albo *Wężyk*).

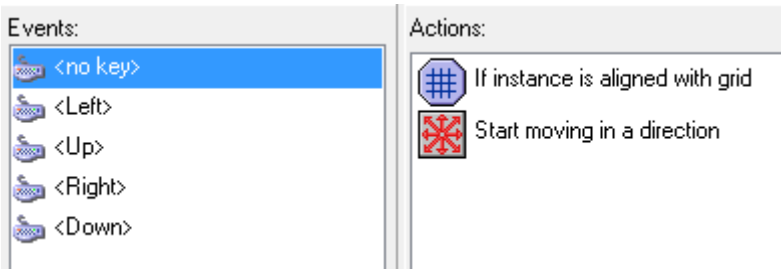
W obiekcie gracz tworzymy zdarzenia przycisku strzałek. Dla każdego zdarzenia dodajemy akcję ruchu w danym kierunku z prędkością 4 (akcja **Move Fixed**). Gdy puścimy klawisz, miś powinien przestać się poruszać w wyznaczonym kierunku. Używamy do tego zdarzenia o nazwie **<no key>** (brak klawisza), w którym dodajemy akcję **Move Fixed** z zaznaczonym środkowym klawiszem kierunku (tak jak pokazano z prawej).

Pojawia się jednak mały problem. Labirynt składa się kratek, w niektórych są ściany, a niektóre są puste i można po nich chodzić. Chcemy by nasz miś zawsze stawał na środku kratki, a nie na przykład jedną nogą na jednej, a drugą na drugiej. To znacznie ułatwi poruszanie i trafianie w korytarze. Istnieje jednak dobry test na to czy obiekt siedzi w środku kratki pokoju. Jest to **Check Grid** , który sprawdza czy obiekt jest dobrze ułożony i dopiero wtedy wykonuje następne akcje (na przykład zmiana kierunku ruchu czy zatrzymanie). Nasz miś zatrzyma się lub skręci tylko wtedy, gdy będzie

ładnie ustawiony w środku kratki. Przed każdym zdarzeniem **Move Fixed** musimy dodać ten test, z wartościami:

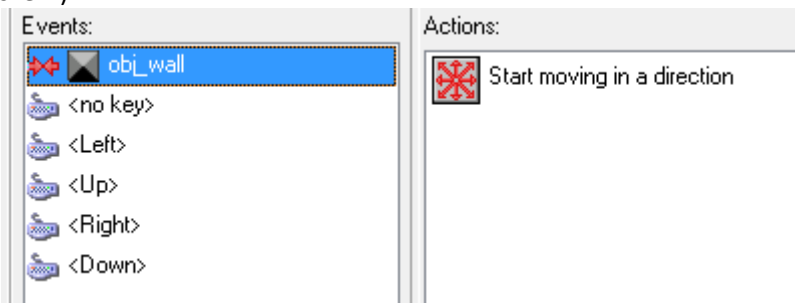


Bo tyle wynosi szerokość i wysokość kratki w labiryncie. Nasze zdarzenia i akcje powinny wyglądać następująco:



Podobne akcje, będą w zdarzeniach strzałek.

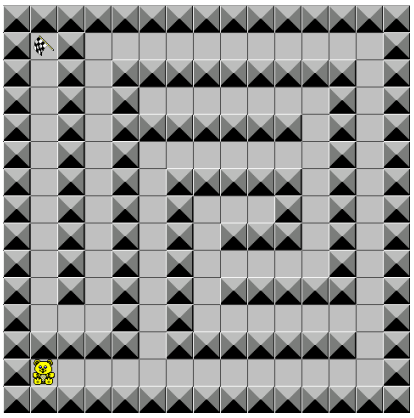
A co jeśli chcemy skręcić w prawo, a po prawej stronie jest ściana? Następuje kolizja gracza ze ścianą, jednak gracz nie może przez nią przejść. Ustawimy w tej kolizji akcję zatrzymania (**Move Fixed**, ze środkowym klawiszem).

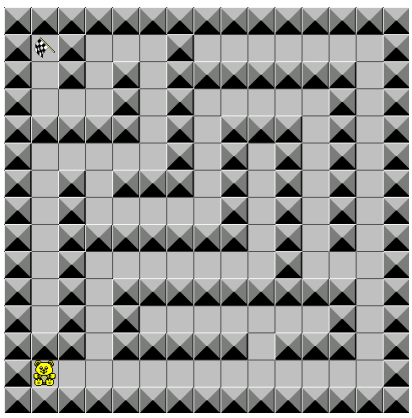


Należy uważać tutaj na jedną rzecz, jeśli obiekt gracza (miś lub coś innego) jest mniejszy niż komórka labiryntu (32x32 piksele) to nie będzie on nigdy wypełniał kratki i testy na to czy dobrze stoi w klatce mogą wypaść źle. Może się więc zdarzyć, że przy kolizji ze ścianą będzie on źle ustawiony i wszystko się zatnie. By tego uniknąć, lepiej pamiętać by obiekty miały wielkość 32x32 lub wyłączyć opcję „Precise Collision Checking" w sprajcie obiektu. U nas na szczęście miś jest dokładnie wielkości 32x32.

Tworzenie pokojów

Skończyliśmy pracę z obiektami, teraz pora na pokoje. Stwórz dwa pokoje, które będą przypominały labirynt. Każdy taki labirynt musi mieć ścianę naokoło by gracz nie wyszedł poza pokój. W każdym pokoju musimy umieścić gracza i flagę (najlepiej z dala od siebie). Przykładowo pokoje mogą być takie:





That was all we had to do in the actions. Now let us create some rooms. Create one or two rooms that look like a maze. In each room place the goal object at the destination and place the person object at the starting position.

Skończone!

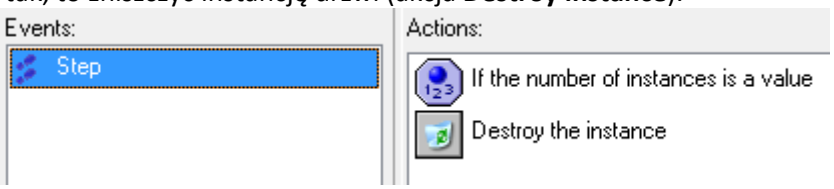
Pierwsza wersja pokoju jest gotowa. Zapisz ją i przetestuj czy wszystko działa dobrze (poruszanie, skręcanie, przechodzenie z pokoju na pokój, restart gry). Jeśli trzeba możesz zmienić prędkość, dodać nowe pokoje lub modyfikować inne rzeczy.

Zbieranie diamentów

Celem gry było zbieranie diamentów, co jest proste do zrobienia. Jak jednak zrobić, by gracz nie mógł wyjść z pokoju bez zebrania wszystkich diamentów? Potrzebujemy nowego obiektu: drzwi. Drzwi będą odgradzały flagę niczym ściana tak długo, aż gracz zbierze wszystkie diemanety. Wtedy dopiero znikną.

Dodaj dwa sprajty dla diamentu i drzwi:

Utwórz na podstawie sprajtów obiekty. Obiekt diament ma proste zachowanie, musimy jedynie zapewnić, że gdy gracz go dotknie to diament zniknie – dodajemy zdarzenie kolizji i akcję zniszczenia instancji diamentu. Drzwi to obiekt, który trzeba wstawić w odpowiednim miejscu (odgradzającym flagę od gracza). Gdy gracz zderzy się z drzwiami, jego ruch musi być zatrzymany. Wystarczy, że obiektowi drzwi nadamy właściwość „Solid" (nieprzenikalne), tak jak zrobiliśmy to ze ścianą. Ponadto drzwi muszą zniknąć po zebraniu wszystkich diamentów. By tego dokonać, drzwi muszą co chwilę sprawdzać (zdarzenie **Step**) czy liczba diamentów w pokoju jest równa 0 (test na ilość instancji diamentu) i jeśli tak, to zniszczyć instancję drzwi (akcja **Destroy instance**).



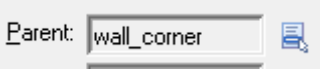
Upiększanie pokoju

Niby wszystko jest na miejscu, ale pokój można ulepszyć. Zamiast jednego obiektu ściany, można wstawić trzy: kawałek odpowiadający za ścianę poziomą, pionową i za róg.

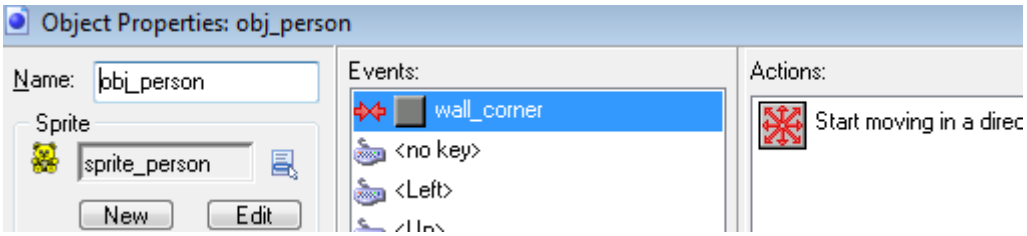
Stwórz trzy sprajty, trzy obiekty z własnością solid. Wymień ściany w pokojach. Powninny wyglądać nieco płynniej I mniej rażąco niż poprzednia wersja:

By pokoje były jeszcze milsze, dodaj obrazek tła i ustaw go jako tło pokoju.

We wszystkim zapomnieliśmy o jeszcze jednej rzeczy. Wejdź do obiektu gracza (miś). Było tam zdaerzenie kolizji ze ścianą. Mamy teraz nowe ściany, ale są ich aż trzy typy. Co można zrobić? Albo zrobić trzy zdarzenia dla każdego typu ściany osobno (proste, ale długie), albo uczynić jeden kawałek ściany nadrzędnym (rodzicem wszystkich ścian). Wystarczy potem badać kolizję dla rodzica, a brane pod uwagę będą tez kolizje z jego dziećmi ścianami.
Ustaw w obiekcie ściany pionowej i ściany poziomej, że ich rodzicem (parent) jest róg ściany:



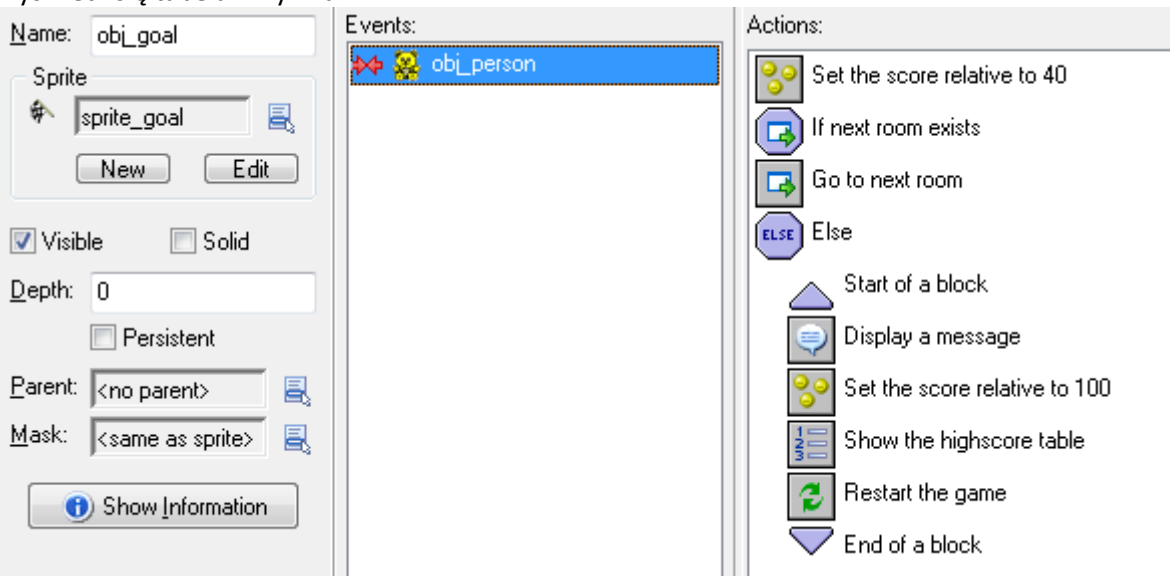
W obiekcie gracz można teraz ustawić kolizję tylko z rogiem ściany, który jest rodzicem wszystkich ścian:



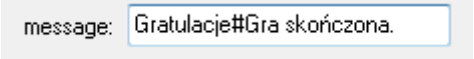
Ponieważ i drzwi są rodzajem ściany i też muszą blokować ruch gracza, również im ustaw, że ich rodzicem jest róg ściany. Dzięki temu zabiegowi załatwiliśmy 4 obiekty w jednym zamachu.

Wynik

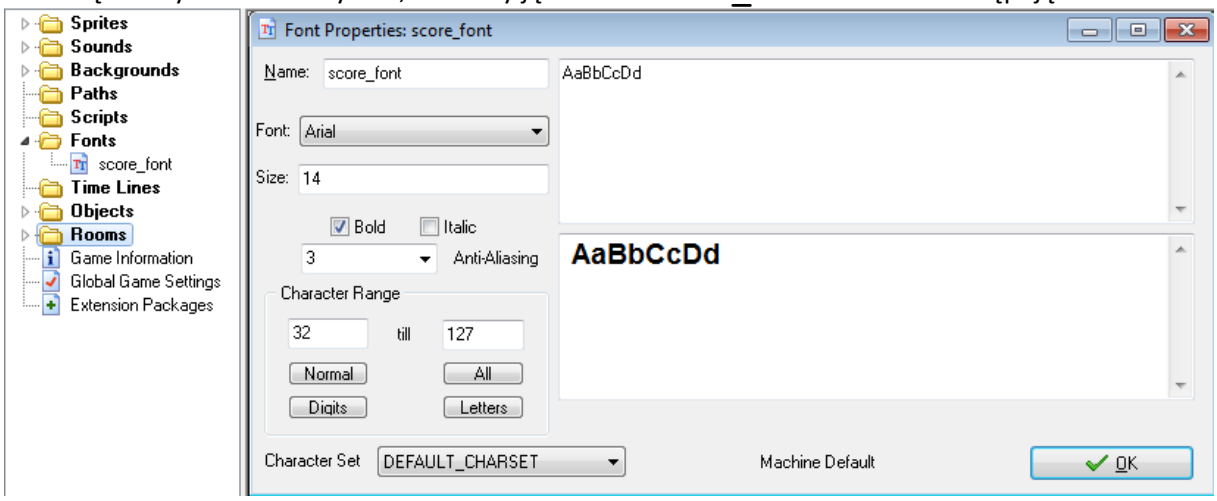
Dodajmy do gry punkty, które będą mierzyły postęp gry. Wystarczy przy zniszczeniu diamentu (zdarzenie) zrobić akcję: dodaj 5 punktów (pamiętaj o Relative). Po ukończeniu poziomu dostajemy dodatkowe 40 punktów, co można dodać jako akcję w zdarzeniu flagi z graczem. Natomiast gdy gra się skończy (nie ma już pokojów) gracz dostanie wiadomość z gratulacjami, dodatkowe 100 punktów i wyświetli się tabela z wynikami:



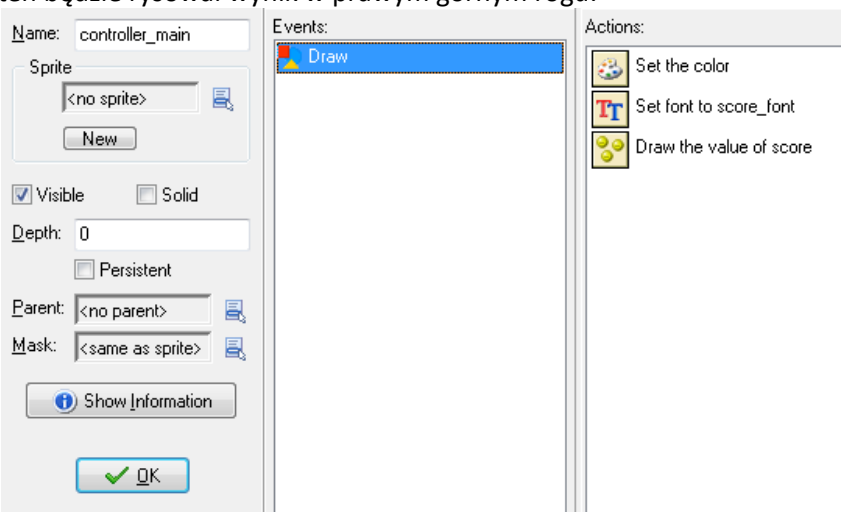
W środku akcji Display Messager (Wyświetl wiadomość) wpisz np.



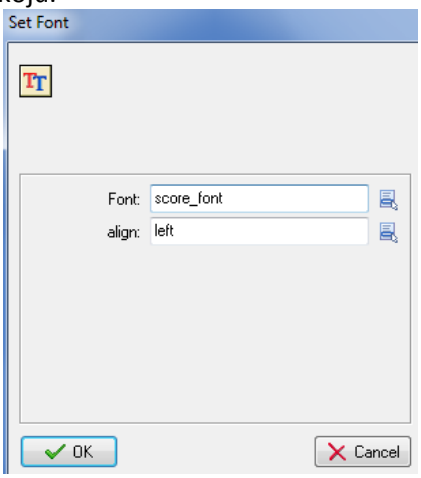
Wynik jest automatycznie wyświetlany w górnym pasku, jednak nie wygląda to ładnie. Stwórzmy czcionkę dla wyświetlania wyniku, możemy ją nazwać **`score_font`** i ustawić następująco:

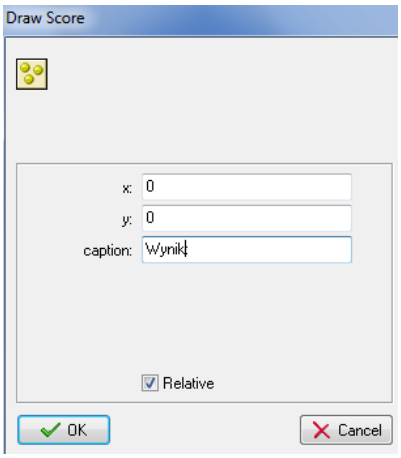


Stworzymy teraz specjalny obiekt kontrolujący np. o nazwie **`controller_main`**, który nie ma sprajta. Obiekt ten będzie rysował wynik w prawym górnym rogu:



Ustawiamy kolor tekstu (np. Czerwony) następnie ustawiamy czcionkę na naszą stworzoną i miejsce napisu w pokoju:





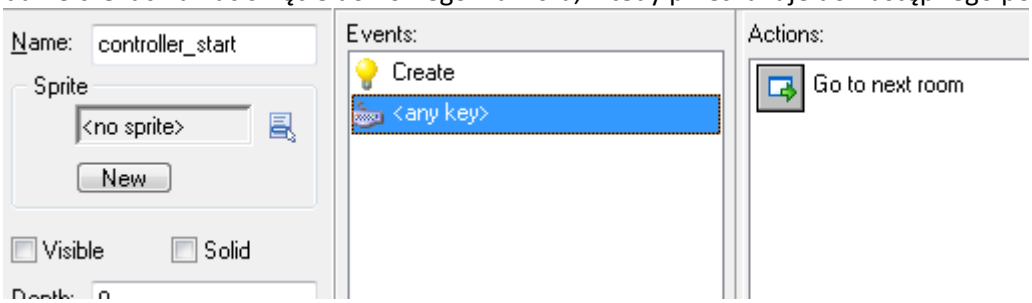
Wrzuć ten obiekt do pokoju tam, gdzie chcesz by pojawiał się wynik:

Ekran startowy

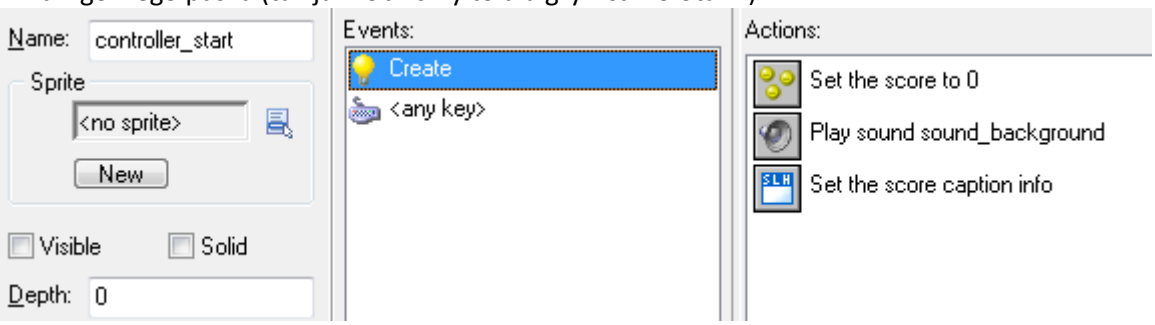
Dobrze jest, gdy gra zaczyna się od ekranu prezentacyjnego, w którym podany jest jej tytuł. Użyjemy do tego pierwszego, specjalnego pokoju, który musisz dodać na początku do spisu pokojów:

Wypełnij go tłem tak jak w pozostałych pokojach. Nastepnię dodaj kolejny sprajt na podstawie obrazka **`start.png`** z folderu Resources i utwórz obiekt na jego podstawie. Dodaj ten obiekt do pokoju startowego w centrum.

Musimy jeszcze dodać obiekt, który czeka na reakcję gracza (naciśnięcie dowolnego klawisza). Nazwijmy go **`controller_start`**, i niech on będzie niewidzialny (odznaczona opcja **Visible**). Ma on za zadanie czekać na naciśnięcie dowolnego klawisza, wtedy przeskakuje do następnego pokoju:



Wykorzystamy też ten obiekt do ustawienia wyniku na 0 (gdy gra jest restartowana) i usunięcia wyniku z górnego paska (tak jak robiliśmy to dla gry z samolotami):



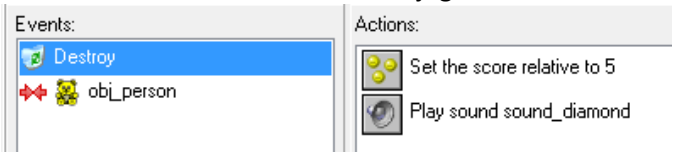
Zauważ, że pojawiła się tu też akcja z dźwiękiem. Dźwiękami zajmiemy się teraz…

Dźwięki

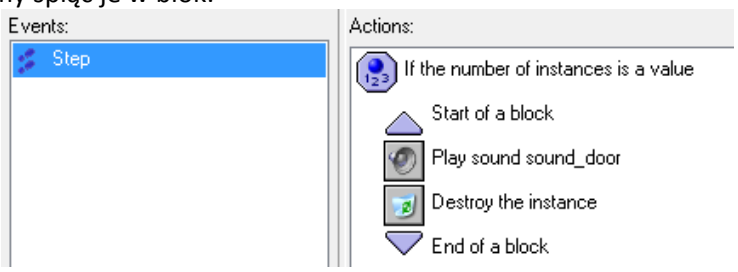
Gra bez dźwięków jest nudna, dodajmy więc kilka. Po pierwsze, muzyka w tle (music.mid). Dodajemy ją do zasobu dźwięków, a następnie jej odtwarzania w obiekcie **`controller_start`** tak jak było pokazane wyżej na obrazku. Zaznacz opcję **Loop**, która zapętli muzykę (będzie odtwarzana w kółko).

Dodajmy też trzy dźwięki do podniesienia diamentu (diamond.wav), otworzenia drzwi (door.wav) i dotknięcia flagi (goal.wav).

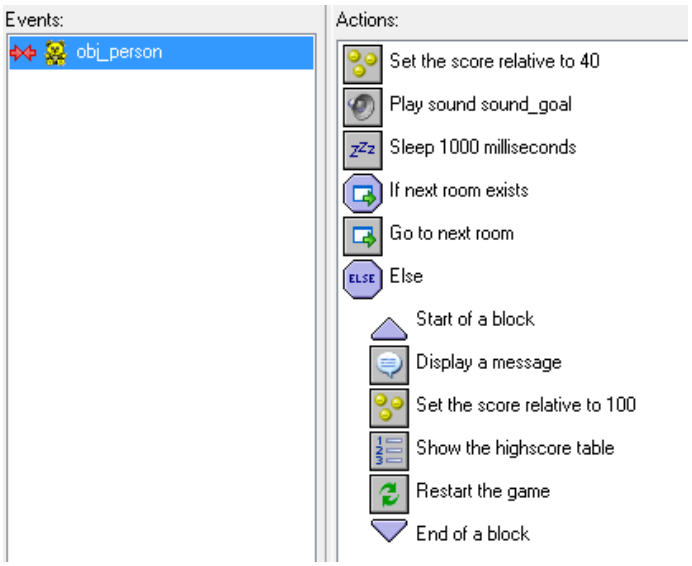
Dźwięk podnoszenia diamentu można dodać w zdarzeniu jego niszczenia:



Dźwięk otworzenia drzwi dodajemy w teście drzwi na liczbę diamentów. Ponieważ test zawiera teraz dwie akcje, to musimy spiąć je w blok:

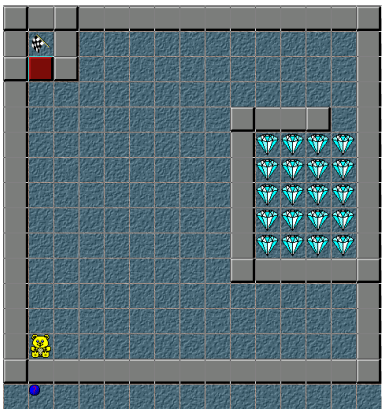


Wreszcie, dźwięk dotknięcia flagi jest odtwarzany w środku flagi w zderzeniu z graczem. Po tym dźwięku trzeba dodać też sekundę pauzy (**Sleep 1000 milisenconds**) po to, aby dźwięk zdążył się cały odtworzyć zanim zrestartuje się gra.

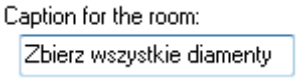


Tworzenie pokojów

Możesz teraz stworzyć pokoje z nowymi ścianami, z diamentami i drzwiami do flagi. Nie zapomnij wrzucić do pokoju startowego obiektu **`controller_start`**, a do pokojów z labiryntami obiektu **`controller_main`**.



Każdemu pokojowi możesz nadawać inny tytuł, by gracz wiedział o co chodzi. Wchodzimy do pokoju, wybieramy zakładkę settings i zmieniamy tytuł np. na:



Potwory i życia

Gra wygląda już ładnie, ale nadal jest za prosta i nudna. Potrzebujemy więcej akcji! Dodajmy wiec potwory. Później dodamy jeszcze inne rzeczy do urozmaicenia.

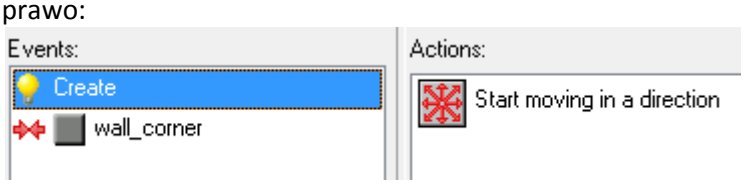
Potwory

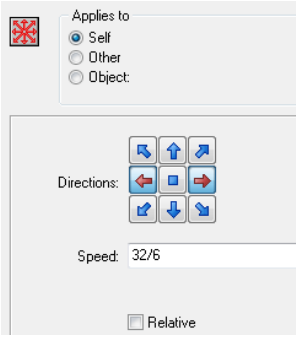
Stworzymy trzy rodzaje potworów. Takie, które poruszają się

- w lewo i w prawo, 
- w górę i w dół, 
- we wszystkich kierunkach. 

Utwórz trzy sprajty i trzy obiekty odpowiadające trzem potworom. Zachowanie potworów jest łątwo zaprogramować. To po prostu obiekt, który porusza się w określonym kierunku, a gdy uderzy w ścianę zmienia kierunek poruszania. Gdy gracz uderzy w potwora to ginie, liczba żyć jest zmniejszana o jeden a poziom jest restartowany. Na początku gracz będzie miał 3 życia.

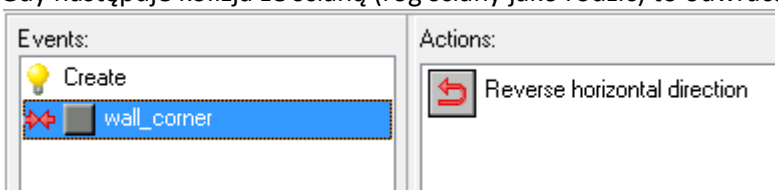
Zajmijmy się teraz potworkiem poruszającym się w liniach poziomych (biały duszek). W zdarzeniu jego kreacji dajemy poruszanie się w lewo lub prawo:





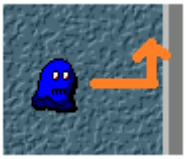
Ustawiamy przy tym sporą szybkość (powyżej 5) , np. 5,3333 to 32/6.

Gdy następuje kolizja ze ścianą (róg ściany jako rodzic) to odwracamy poruszanie w poziomie:



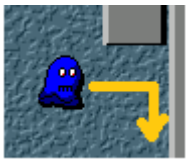
Drugi potworek (pająk) działa prawie tak samo. Musisz jedynie zamienić poruszanie w lewo i prawo, na górę i dół. W kolizji zaś na odwrócenie pionowego (vertical) kierunku.

Trzeci potworek (ciemny duch) jest trochę trudniejszy w obsłudze. Zacznie poruszać się pionowo lub poziomo (zaznaczamy cztery strzałki). Przy zderzeniu ze ścianą próbuje iść w lewo lub w prawo, a jeśli nie może to odwraca kierunek (idzie do tyłu).



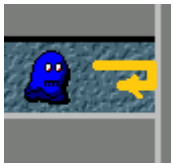
(zderzam się, skręcam w lewo)

(kierunek zmienia się o +90°)



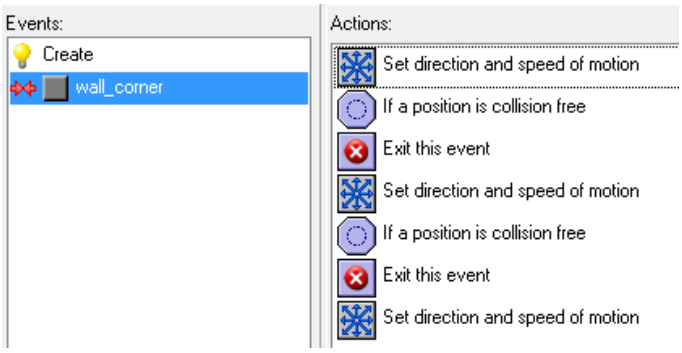
(zderzam się, nie mogę w lewo, to skręcam w prawo)

(kierunek zmieniony +90° zmienia się jeszcze o +180°)

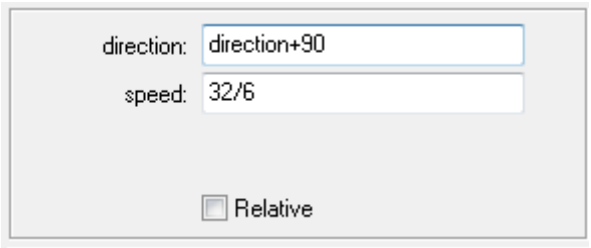


(zderzam się i nie mogę iść w lewo ani prawo, to cofam się)

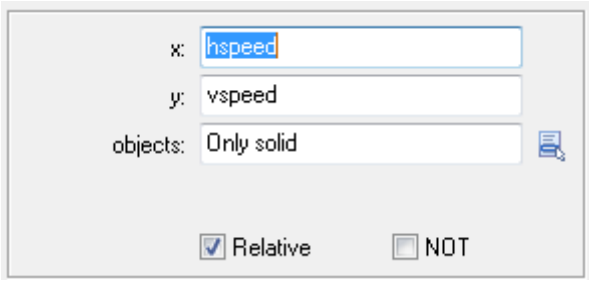
(kierunek zmieniony w sumie o +270° zmienia się o -90°)



Pierwsza akcja (niebieskie strzałki), zmienia kierunek ruchu o 90 stopni przeciwnie do wskazówek zegara.

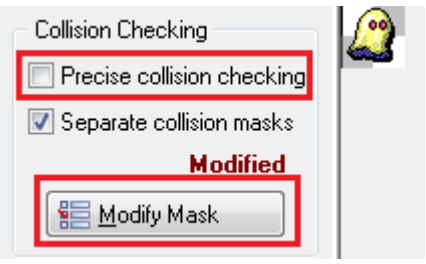


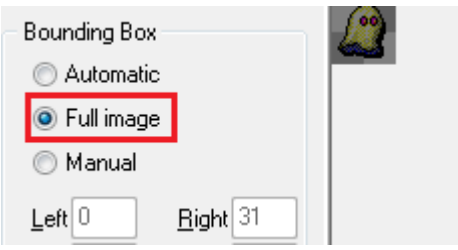
Następnie druga akcja bada, czy można się w tym kierunku poruszać (czy pozycja jest wolna od kolizji?):



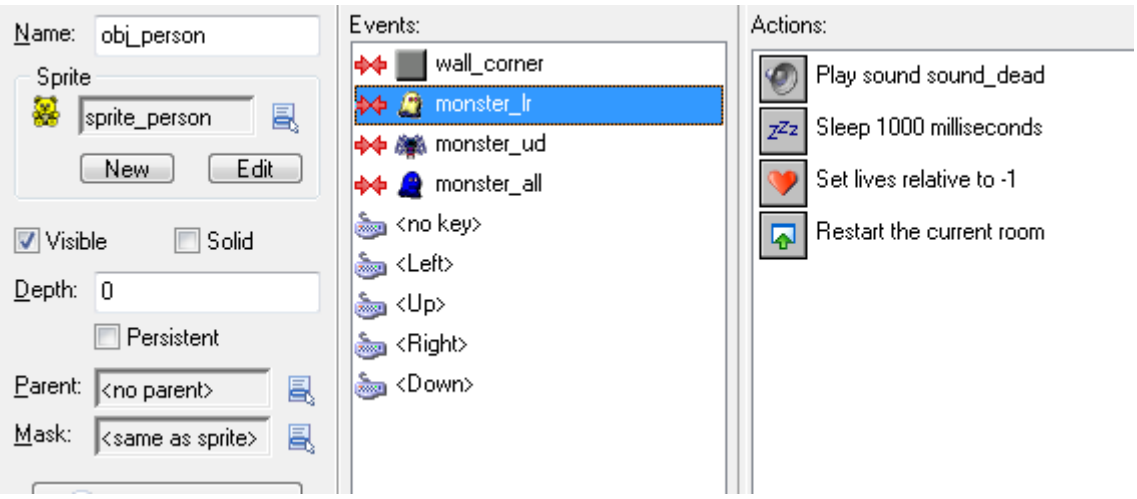
Jeśli tak, to zdarzenie się kończy (**Exit this event**) i potwór swobodnie idzie dalej. Jeśli jednak nie to w kolejnej zmianie kierunku dodajemy do **direction+180**. Następnie znów sprawdzamy czy droga wolna. Jeśli nie, to ostatnia zmiana kierunku to **direction-90**.

By upewnić się, ze kolizje będą przebiegały prawidłowo we wszystkich sprajtach potworków odznacz precyzyjne kolizje oraz wejdź do modyfikacji obrazka (**Modify Mask**) i ustaw wartość **Bounding Box** na **Full Image** (zderzenie z potworkiem to zderzenie z całą jego kratką).

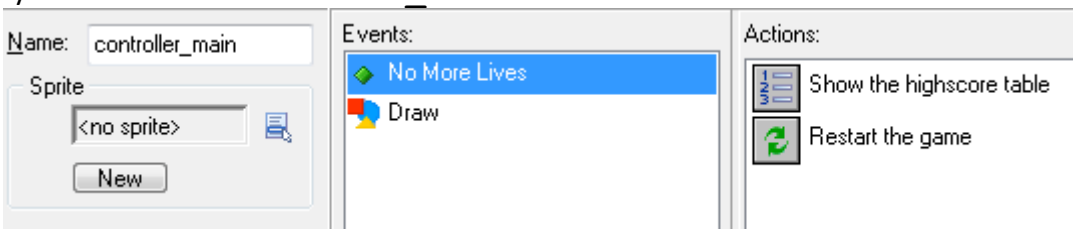




Gdy gracz zderza się z potworkiem dzieje się kilka akcji: odgrywany jest jakiś dźwięk krzyku (dead.wav – musisz dodać do zasobów), następnie sekundę czekamy, aż cały dźwięk się odegra, zmniejszamy liczbę żyć o 1 i restartujemy pokój. Wygląda to następująco:

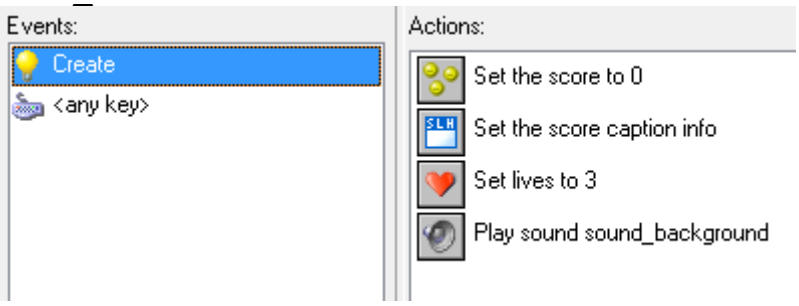


Te cztery akcje, należy ustawić dla pozostałych dwóch potworków w kolejnych zdarzeniach. Zwróć również uwagę, że kolejność akcji jest ważna – gdyby restartowanie pokoju było wyżej to następne akcje by się już nie wykonały bo wyszliśmy z pokoju. A co się dzieje, gdy brakuje żyć? W obiekcie **`controller_main`** trzeba dodać zdarzenie:

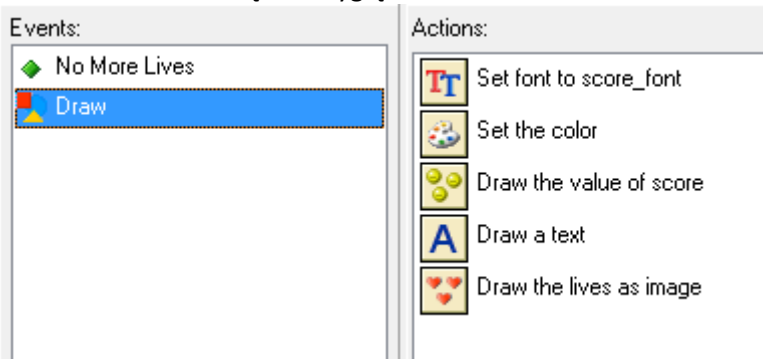


Życia

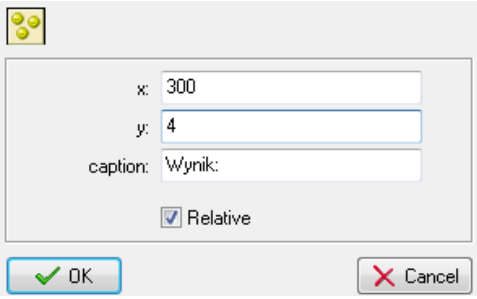
Zaczęliśmy już zarządzać życiami gracza. Na początku trzeba ustawić ich ilość na 3. Robimy to w obiekcie **`controller_start`**:

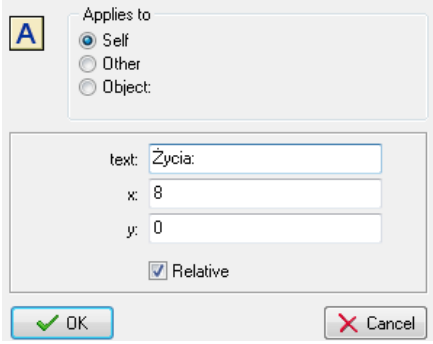


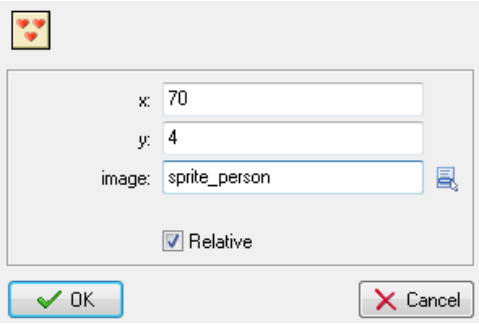
Życia zmniejszają się po zderzeniu z potworkami, a gdy ich braknie to gra jest restartowana. To już zaprogramowaliśmy. Pozostaje nam dodanie wyświetlania liczby żyć. Podobnie jak to było w samolotowej strzelance wyświetlimy życia w formia małych obrazków (można wykorzystać obrazek misia). Akcja rysowania w kontrolerze będzie wyglądała tak:



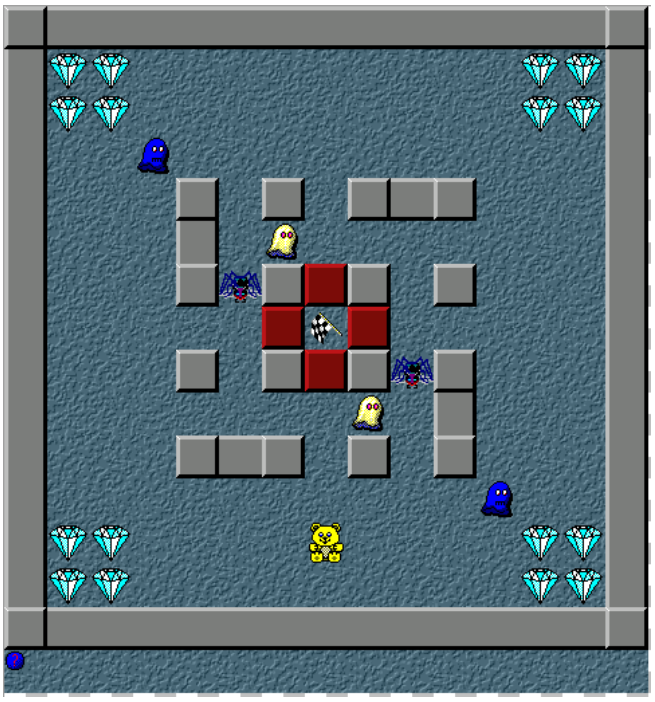
Trzy ostatnie z tych pięciu akcji należy uzupełnić następująco:







Stwórz jakiś ciekawy pokój, w którym potworki będą mocno przeszkadzały w zdobywaniu diamentów np.



Zapisz grę i przetestuj ją.

Bomby, bloczki, dziury

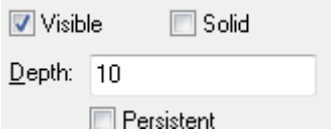
Uatrakcyjnimy teraz grę o kolejne elementy.

Bomby

Dodamy bomby oraz zapalniki, które służą do ich wysadzania. Pomysł jest taki, że gdy gracz dotknie zapalnika, wszystkie bomby wybuchną i zniszczą wszystko co z nimi sąsiaduje. Mogą niszczyć ściany, potwory. Potrzebujemy trzech rzeczy: zapalnika (trigger), bomby (bomb) i eksplozji (explosion). Stwórz 3 sprajty na podstawie obrazków dla wyżej wymienionych obiektów:

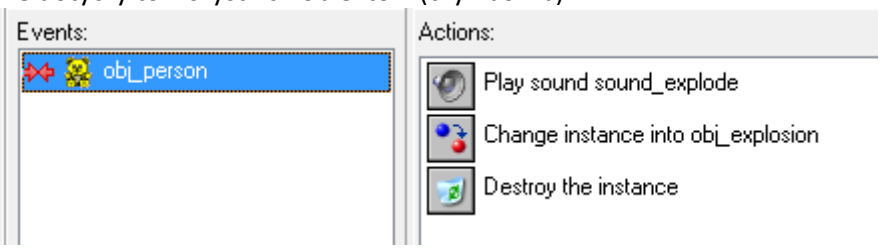
Następnie stwórz obiekty na podstawie tych sprajtów. Pora zastanowić się jak zachowują się te obiekty.

Działanie bomby jest proste – po prostu siedzi i nic nie robi. By potworki i gracz przechodzili nad bombą zamiast pod nią ustaw głębokość jej położenia na 10:

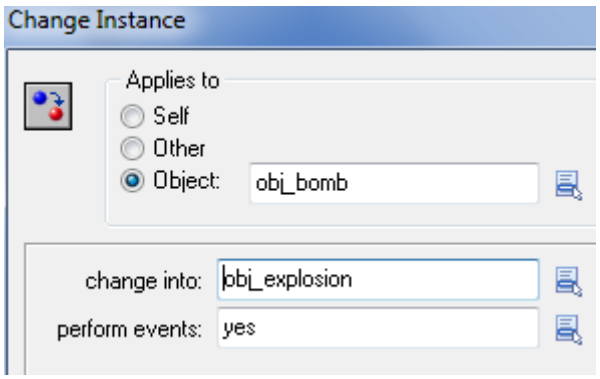


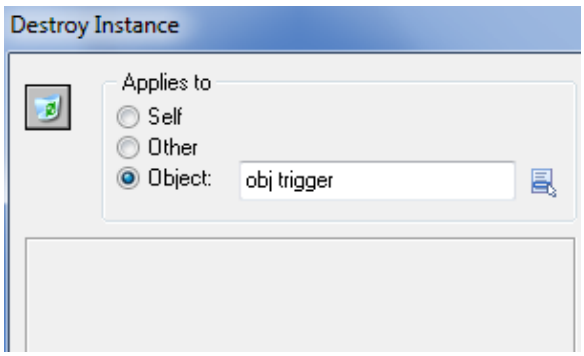
Wszystkie inne obiekty mają głębokość 0, więc będą rysowane nad bombą.

Zapalnik też jest dość prosty w działaniu. Gdy zderzy się z graczem (kolizja) zamienia on wszystkie bomby w eksplozje. Możemy to zrobić za pomocą akcji zamieniającej jeden obiekt w drugi. Musimy też zaznaczyć, że dotyczy to wszystkich obiektów (czyli bomb):



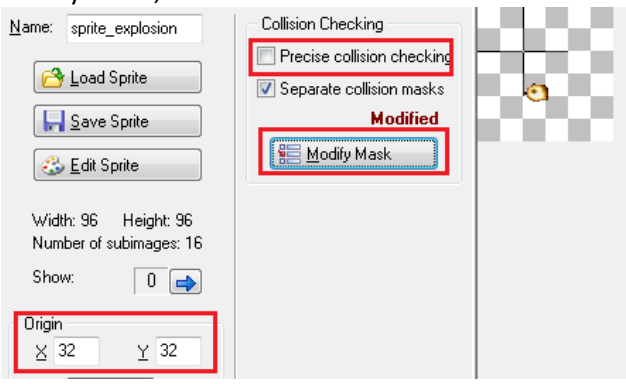
Pierwsza akcja to odegranie dźwięku (możesz dodać dźwięk eksplozji a potem tę akcję - to jest łatwe). Druga akcja zamienia wszystkie bomby w eksplozje, a trzecia wszystkie zapalniki usuwa z planszy.



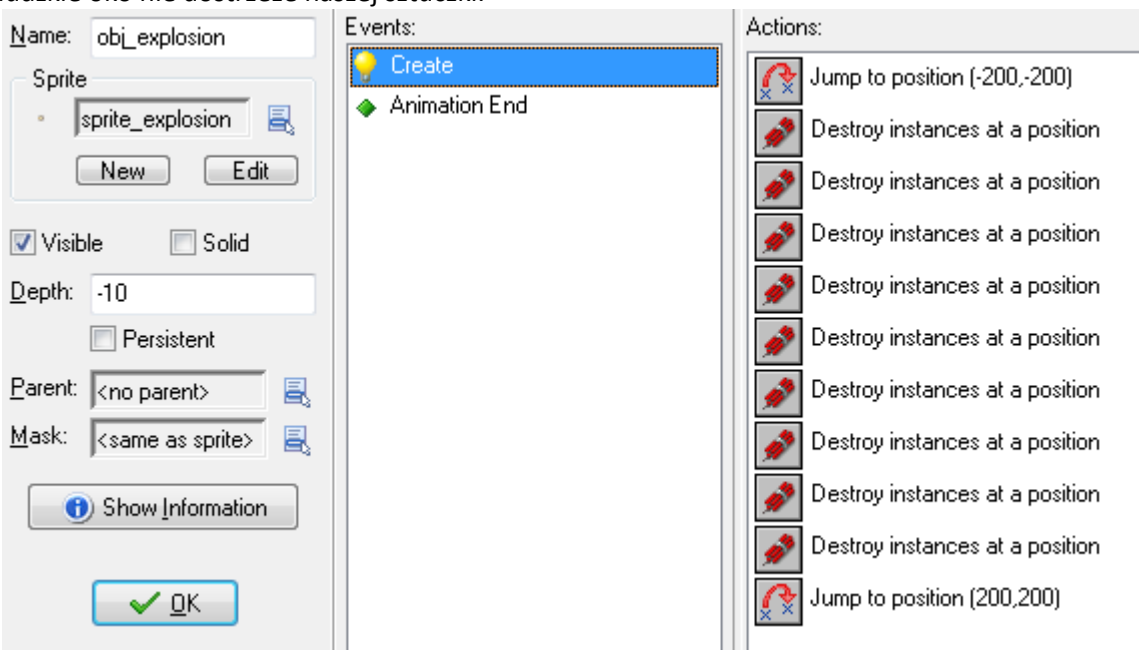


Eksplozja to obiekt ukazujący animację eksplozji. Po tej animacji obiekt się usuwa, ale usuwa też wszystko czego dotyka (8 sąsiadujących kratek i kratka w której była bomba). Do tego trzeba trochę pracy:

- Upewnij się, że sprajt eksplozji ma wyłączone **Precise colission checking** oraz że pod guzikiem **Modify mask** ma ustawione Bounding box na **Full image**. Dodatkowo punkt zaczepienia (origin) musi być ustawiony na 32,32:

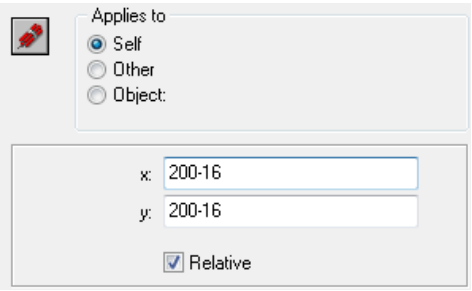


- Wracamy ze sprajta do obiektu eksplozji. W zdarzeniu tworzenia obiektu (czyli w momencie gdy eksplozja pojawia się na planszy) usuwamy wszystkie możliwe obiekty z okolicy. Ale chwileczkę! Wtedy usuniemy też eksplozję zanim cała animacja się odpali. Musimy więc eksplozję na chwilkę (dosłownie milisekundy) przesunąć z dala od miejsca eksplozji, następnie zniszczyć wszystkie obiekty w miejscu eksplozji i przesunąć eksplozję z powrotem w swoje miejsce, by odpalić animację do końca. Wszystko to dzieje się w ułamku sekundy, ludzkie oko nie dostrzeże naszej sztuczki:



Uwaga: w akcjach **Jump to position** zaznacz opcję **Relative**. Natomiast 9 akcji Destroy instances at position powinno być uzupełnionych wartościami:

200-16,200-16	200,200-16	200+16,200-16
200-16,200	200,200	200+16,200
200-16,200+16	200,200+16	200+16,200+16

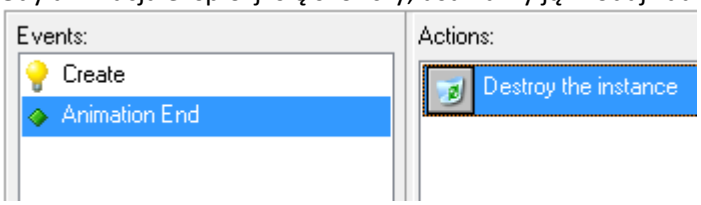


Liczbę 200 dorzucamy wszędzie, bo eksplozję na początku przenieśliśmy o -200,-200 pikseli.
Liczba 16 z różnymi znakami mówi, w jakiej kratce Usuwamy instancję.

Przykładowo by trafić w kratkę na lewo od eksplozji i usunąć w niej potwora musimy przesunąć się o 16 pikseli w lewo (-16) i 0 pikseli górę lub dół.

Nigdy nie ustawiaj zapalnika zbyt blisko bomby bo wtedy zginie i gracz. Bomba też nie powinna leżeć przy ścianie bocznej pokoju lub przy fladze. Te obiekty nie mogą być niszczone.

- Gdy animacja eksplozji się skończy, usuwamy ją. Dodaj zdarzenie i akcję:

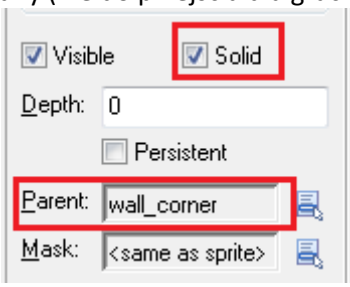


- Ustaw głębokość eksplozji na -10. Będzie ona animowana nad wszystkimi obiektami a nie pod nimi. Dziwnie by wyglądało gdyby wybuch był pod ścianą.

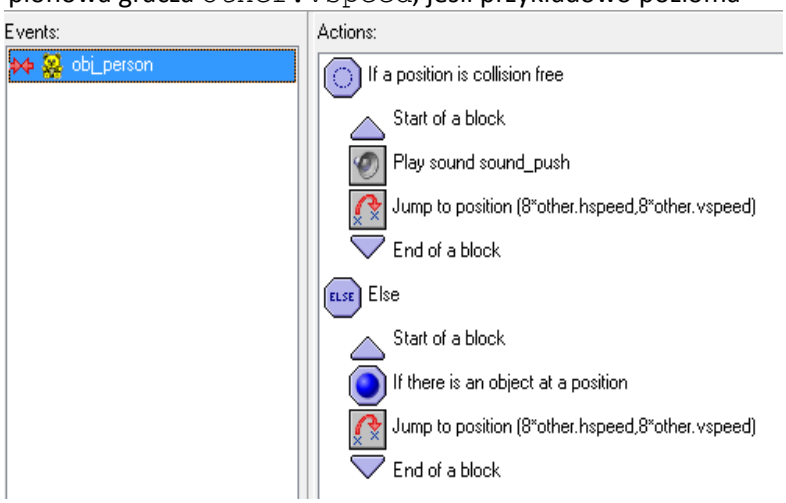
Bloki i dziury

Stwórzmy teraz coś, co pozwoli na robienie trudniejszych poziomów i zmusi graczy do główkowania. Stworzymy kamienne bloki, które mogą być popychane i przesuwane przez gracza. Zrobimy też dziury, przez które gracz nie może przejść, chyba że wrzuci do nich blok. Bloki będą więc potrzebne do tworzenia nowych przejść, ale też do zatrzymywania potworów. Dodaj sprajta dla bloku i dziury:

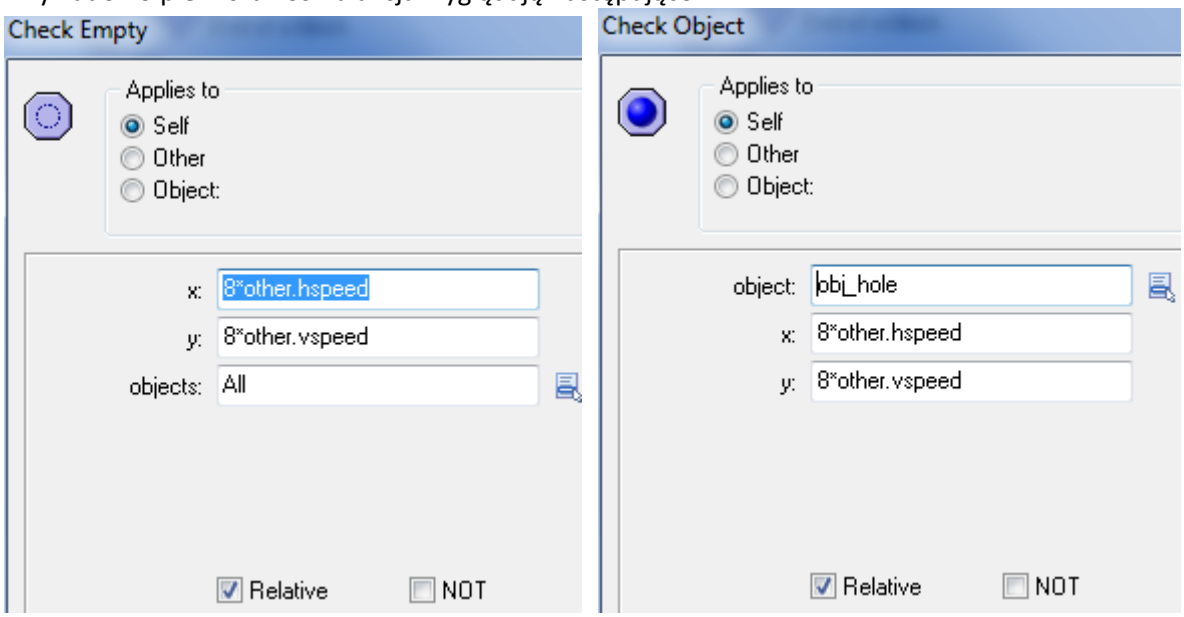
Stwórz obiekty na ich podstawie i obu obiektom zaznacz opcję **Solid** oraz rodzica (**Parent**) róg ściany. Dzięki temu będą traktowane jak ściany (nie do przejścia dla gracza i potworów).



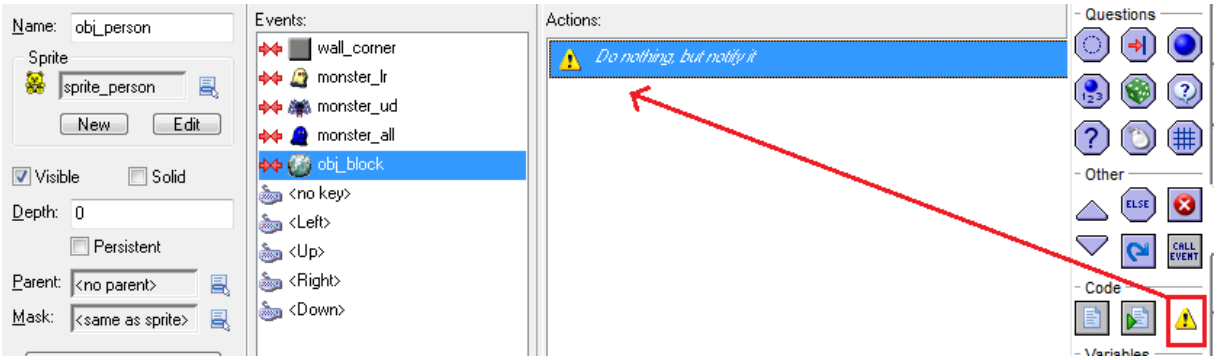
Zastanówmy się nad zachowanie bloku. Musi on podążać za ruchem gracza gdy jest popychany. Gdy następuje kolizja z graczem blok sprytnie wykryje z której strony jest popychany (sprawdzi jaka jest prędkość pozioma `other.hspeed` i pionowa gracza `other.vspeed`, jeśli przykładowo pozioma jest dodatnia, to znaczy, że gracz idzie w prawo i blok musi przesunąć się w prawo). Blok więc musi jedynie sprawdzić czy pozycja w labiryncie o współrzędnych `8*other.hspeed`, `8*other.vspeed` jest pusta. Jeśli jest pusta to przesuwamy tam blok. Jeśli nie jest pusta to musimy jeszcze sprawdzić czy jest tam dziura. Jeśli jest, to wrzucamy blok do dziury.



Wszystkie akcje przedstawiono wyżej na obrazku. Wszystkie muszą mieć pozaznaczane **Relative**. Przykładowo pierwsza i ósma akcja wyglądają następująco:

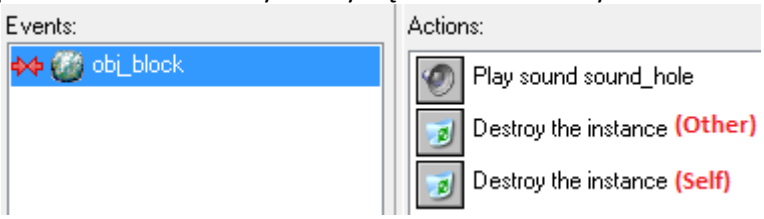


Jest jeszcze jeden problem. Ponieważ blok będzie zachowywał się jak ściana (bo róg ściany jest jego rodzicem) to będzie on blokował ruch gracza (tak jak inne ściany). By ruch nie był blokowany dodamy w obiekcie gracza kolizję z blokiem, a jako akcję dodamy tylko akcję „komentarz", która tak naprawdę nic nie robi.



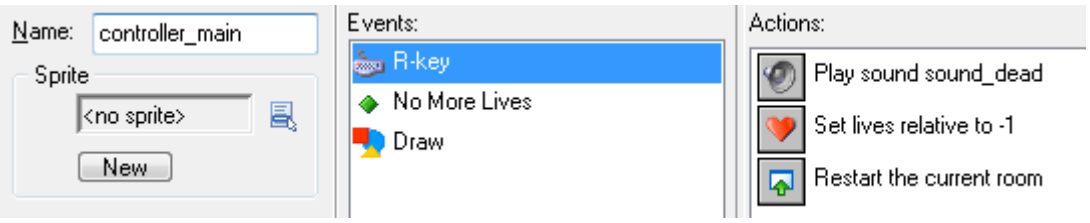
Innymi słowy, kolizja z blokiem „nadpisuje" kolizję ze ścianą. Nie chcemy by blok zachowywał się zupełnie tak samo jak jego rodzic (czyli ściana) więc dodajemy akcję, która zastępuje kolizję ze ścianą.

Dziura jest bardzo prosta w obsłudze. Gdy zderzy się z blokiem niszczy siebie i blok:

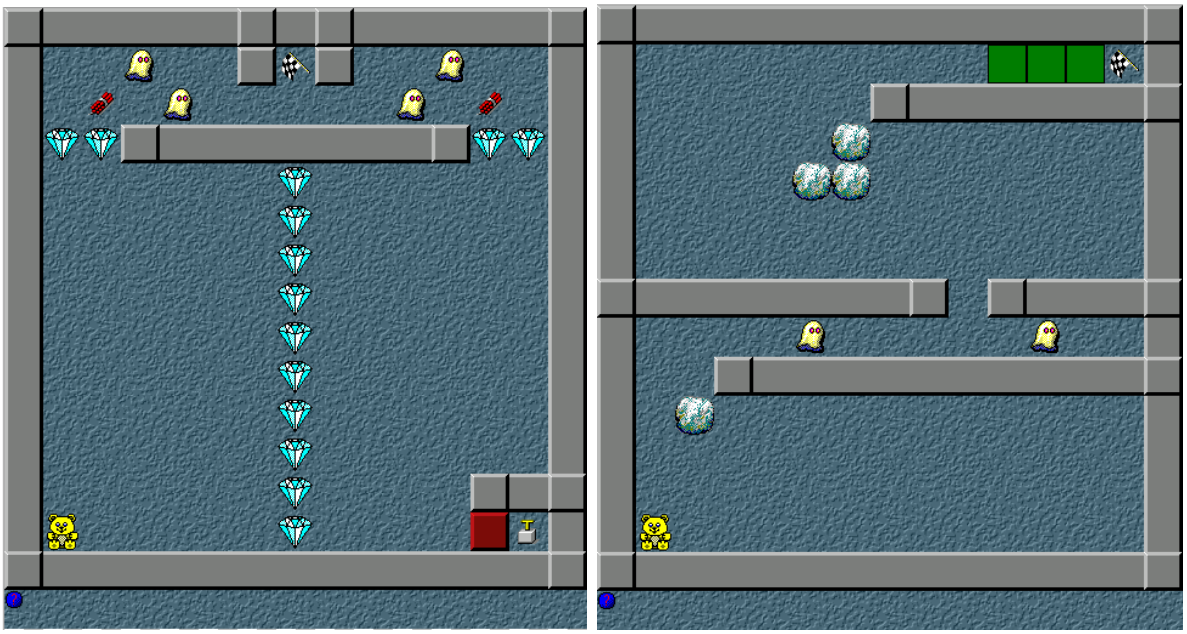


Do wrzucania bloku do dziury i przesuwania bloku (jak było widać wyżej), można dodać dźwięki, co nie powinno stanowić dużego problemu.

Wreszcie, ostatnia uwaga. Może się zdarzyć, że gracz zablokuje się blokami i nie będzie miał drogi wyjścia. Potrzebujemy zatem możliwości restartu poziomu (z zabraniem życia). By ręcznie zrestartować poziom, gracz będzie musiał nacisnąć guzik R na klawiaturze (R jak Restart). Działanie tego guzika można oprogramować w kontrolerze pokoju:



I gotowe! Wystarczy dodać nowe ciekawe pokoje, które zmuszają gracza do kombinowania, np.:



Kilka uwag końcowych

- Pamiętaj by poziomy projektować od najłatwiejszych do najtrudniejszych. W początkowych poziomach nie używaj wszystkich obiektów. Niech nowe obiekty, potwory pojawiają się później. To wprowadzi element niespodzianki do gry i gracz będzie miał motywację by odkrywać kolejne poziomy.
- Tak projektuj poziomy, żeby wymagały myślenia (jak dojść do drzwi, jak przesunąć bloki, jak ominąć potworki). Szczególnie dalsze poziomy powinny być trochę trudniejsze.
- Gra nie może szybko się kończyć. Powinna mieć kilkadziesiąt poziomów, by zapewnić rozrywki na godzinę lub nawet kilka godzin.
- Gracze nie lubią przechodzić gry za jednym zamachem. Na szczęście GameMaker ma możliwość zapisywania gry (naciśnięcie F5) i załadowania zapisanej gry (klawisz F6). Umieść tę informację (oraz informacje o zasadach gry i innych klawiszach) w dokumentacji gry czyli **Game Information**  (tak jak robiliśmy to w grze z Klaunem).