KropekFX Interface 1.5

Release 151

Oficjalna specyfikacja

1. Klasy

1.1. KDrawList

KDrawList to klasa reprezentująca dane rysowania. Korzystają z niej modele, aby nie duplikować za każdym razem danych rysowania.

destructor Destroy();

Dealokuje dane rysowania, niszczy klasę.

function inUse: bool;

Zwraca prawdę jeśli lista rysowania jest ciągle w użyciu przez jakieś modele.

1.2. KRotatable

Klasa ta zawiera metody służące do działań na obiektach które mają pozycję i które można obracać.

procedure StackUp;

zapamiętuje w wewnętrzym buforze aktualną pozycję obiektu.

procedure StackDown;

przywraca pozycję z wewnętrznego bufora. Generalnie ruch w kolizjach realizuje się tak: {StackUp->ustawienie potencjalnej nowej pozycji obiektu->sprawdzenie kolizji->jeśli jest kolizja to StackDown}. To tylko zalecenia, po prostu przydatne rzeczy.

procedure applyRotation(n: float; axis: r3dp);

obraca o n stopni wokół osi definiowanej przez wektor axis.

procedure SetPosition(x,y,z: float);

procedure SetPosition(p: r3dp);

ustawia daną pozycję obiektowi

position: r3dp;

zmienna ta zawiera aktualną pozycję. Można ją modyfikować(jest to property).

procedure Zero;

ustawia pozycję (0,0,0), zeruje macierz obrotów.

Up: r3dp;

zawiera aktualny wektor lokalnej osi Y. Może przyjmować wartości nieznormalizowane.

Forward: r3dp;

zawiera aktualny wektor lokalnej osi Z. Może przyjmować wartości nieznormalizowane.

Lookat: r3dp;

przyjmuje współrzędne świata na które należy wykierować lokalną oś Z. Nie można z niej czytać!

procedure moveFoward(n: float);

procedure moveBack(n: float);

procedure moveUp(n: float);

procedure moveDown(n: float);

przesuwa obiekt o n jednostek w zadanym kierunku. Procedura kieruje się lokalnym układem współrzędnych.

1.3. KCamera (dziedziczy po KRotatable)

*C*urrent: bool;

jeśli ma wartość prawda, to przy każdej zmianie pozycji czy obrotu kamera zostanie automatycznie przesłana do OpenGL jako aktualna. Ustawienie jej wartości prawda powoduje automatyczne przesłanie jej.

procedure FetchToGl;

wysyła kamerę do OpenGLa

1.2. KModel (dziedziczy po KRotatable)

drawData: KDrawList;

aktualna lista rysowania

name: str;

nazwa obiektu

procedure Render;

renderuje obiekt

procedure SetDrawList(dl: KDrawList);

jedyna słuszna procedura do ustawiania aktualnej listy rysowania

procedure ResetDrawList;

jedyna słuszna procedura do nilowania aktualnej listy rysowania. Odłącza listę rysowania od modelu i ustawia jej wartość nil.

collisioner: fCollidable;

Jakaś klasa kolizji.

1.3. KModelSet *(dziedziczy po KRotatable)*

Czasem zachodzi potrzeba umieszczenia n obiektów w wspólnym układzie współrzędnym, np. żołnierza + karabin + ogień z lufy. Taką abstrakcję zapewnia KModelSet. Najpierw wykonywane są translacje i rotacji opisane przez KModelSet, a dopiero później indywidualnych obiektów. Lista KModelSet działa jak tablica array, jednakże można do niej pisać pod każdy indeks pod jaki ci się zachce. Tablica alokacje pod nowe indeksy zautomatyzuje. Tablica będzie sama dbać o swoją wielkość – jeśli na końcu jest nil to go odetnie. Proste i smaczne.

ModelCount: ubMW;

zwraca ilość modeli w zestawie

Models: array[0..n] of KModel;

tablica modeli. Kolejne indeksy do których tworzone są odwołania zostaną wygenerowane automatycznie, to taka 'tablica automatyczna'. Tabela automatycznie zwróci nil jeśli zaadresowany zostanie niezaalokowany wcześniej.

procedure Render;

wrzuć na dechę translacje i rotacje KModelSet i wyrenderuj wszystkie modele z listy.

collisioner: fCollidable;

naprawdę, to jest robione tylko dla wygody użytkownika.

1.2. KSystemManager

procedure RefreshInput;

odświeża stan urządzeń wejścia.

procedure SystemBusyLoop;

czeka aż piknie timer. W tym czasie przetwarza wiadomości.

procedure setTimer(ms: ubMW);

ustawia timer na ileśtam milisekund. Można zmieniać nawet przy włączonym timerze, w tym wypadku procedura automatycznie ubije aktualny timer;

Hint: do zabijania timera bez tworzenia nowego użyj setTimer(0)

function userRequestedQuit: bool;

zwraca prawdę jeśli okienko GUI było zamykane lub program dostał wiadomość zamykającą.

function GetValue(val: ubMW): ubMW;

zwraca wartość systemową. Dla następujących wartości val zwrócone zostaną:

KGETVAL_RC	Kontekst renderingu
KGETVAL_DC	Kontekst urządzenia
KGETVAL_HWND	Uchwyt do okna
KGETVAL_HINST	Instancja aplikacji (hInstance)

PerfMgr.setAntialiasing(aa: ubMW);

ustawia antialiasing na podaną wartość. Możliwe wartości to:

KPERFMGR_ANTIALIASING_NONE	Antialiasing wyłączony
KPERFMGR_ANTIALIASING_LOW	Niska jakość
KPERFMGR_ANTIALIASING_HIGH	Wysoka jakość

procedure PerfMgr.RegisterTime;

procedura do liczenia FPSów

function PerfMgr.GetCallPerSecond: float;

zwraca ilość wywołań RegisterTime() na sekundę. Do liczenia FPSów.

Keyboard.isKeyDown(key: ub16): bool;

jako parametr bierze oznaczenia klawisza VK_*(moduł Windows) i zwraca jeśli jest on WCIŚNIĘTY

Keyboard.isKeyPressed(key: ub16): bool;

jako parametr bierze oznaczenie klawisza VK_*(moduł Windows) i zwraca jeśli był wciśnięty i następnie zwolniony.

Mouse.mode

aktualny tryb myszy. Może być jednym z dwóch:

KMOUSEMODE_FREE: Dostępne są pole:

Mouse.X, Mouse.Y: ubMW; współrzędne myszy od startu okienka

KMOUSEMODE_DELTA: Dostępne są pola:

Mouse.dX, Mouse.dY: bMW; opisujące przesunięcie od środka ekranu. W trybie KMOUSEMODE_DELTA mysz co odświeżenie stanu wejść jest automatycznie przesuwana na środek ekranu.

Mouse.mbuttons: MouseButtonStatuses;

stany klawiszy myszy.

2. Instrukcje ogólne

function getDrawList3DS(path: str; scale: r3dp): KDrawList;

Zwraca listę rysowania wygenerowaną z podanego pliku 3DS przeskalowanego o scale.

Initialize(wndname: str; x,y,w,h: ubMW; fov, nearclip, farclip: float);

Inicjuje silnik. Tworzy okno o tytule wndname na pozycji (x,y) o wielkości (w,h). Fov określa pole widzenia w perspektywie, nearclip i farclip opisują płaszczyzny obcinania.

Finalize;

Zamyka silnik.

SetMode(aspect, data: ubMW);

Ustawia tryb pracy silnika. Możliwe wartości to:

KINIT_FILLMODE (domyślnie włączone)	KINIT_ENABLE: Do rysowania wykorzystywane będzie wypełnienie
	KINIT_DISABLE: Rysowanie będzie odbywać się w trybie wireframe

KINIT_LIGHT (domyślnie wyłączone)	KINIT_ENABLE, KINIT_DISABLE: Zarezerwowane.
KINIT_CULLING (domyślnie włączone)	KINIT_ENABLE: Włącza backface culling
	KINIT_DISABLE: Wyłącza backface culling
KINIT_ZBUFFER (domyślnie włączone)	KINIT_ENABLE: Włącza zbuffer
	KINIT_DISABLE: Wyłącza zbuffer
KINIT_COLOR_MODELS (domyślnie włączone)	KINIT_ENABLE, KNIT_DISABLE: Zarezerwowane

SetParam(aspect: ubMW; data: lp);

Podaje silnikowi parametr do kontroli zachowania. Jeśli data wskazuje na strukturę to po podaniu można ją zwolnić. Możliwe wartości i oczekiwane dane to:

KINIT_CLEARBUFFER	Dane to wskaźnik na crgb zawierającą kolor na który czyszczony będzie bufor.

KFXStartRender;

Czyści bufor, rozpoczyna fazę przyjmowania danych do wyświetlenia

KFXStartAccepts;

Rozpoczyna przyjmowanie danych 3D do renderingu

KFXEndAccepts;

Renderuje dane do bufora, kończy przyjmowanie danych 3D do renderingu

KFXEndRender;

Renderuje wszystkie dane do bufora, wykonuje flip.

3. Kolizje

Podstawową jednostką kolizji jest obiekt kolizji. Obiekt kolizji dziedziczy po fCollidable:

fCollidable

constructor Create(p: KPositionable);

tworzy instancję. Jako rodzic zapisana zostanie dana klasa KPositionable. Jeśli obiekt nie ma mieć rodzica, podać nil.

function collideAgainst(n: fColliable): bool;

jeśli dany obiekt koliduje z n, zwracana jest prawda.

Jest sobie parę klas którym można robić kolizję. Te klasy to:

3.1. fCylinder

walec

procedure FromDrawList(p: KDrawList);

inicjuje wewnętrzne parametry cylindra przy użyciu drawlisty. Najczęściej korzystać się będzie z drawlisty danego modelu. Po wywołaniu tej procedury można dealokować drawlistę.

3.2. fGroundPlane

półprzestrzeń o granicy w płaszczyźnie XZ.

Y: float;

Definiuje pozycję płaszczyzny granicznej.

floor: bool;

Jeśli prawda, to półprzestrzeń będzie iść w dół osi Y, w przeciwnym wypadku w górę.

3.3. fCollisionSet

zestaw kolizji traktowany jako jeden obiekt.

procedure AddItem(a: fCollidable);

dodaje obiekt do listy

procedure ResetItems;

czyści listę.

4. Generalne zalecenia

4.1. Korzystaj z destruktorów

To naprawdę niesamowicie ważne aby korzystać z destruktorów. Obiekty alokują różne rzeczy za plecami a przypisanie im po prostu nila to wstęp do katastrofy[czytaj wycieku pamięci].