# S18T – Terminalowy silnik gry

Dokumentacja

## Spis treści



# 1 Wstęp

Dokument dotyczy silnika gry S18T, wersja 1.0 dev. Biblioteka umożliwia tworzenie gier w terminali, wspiera systemu Unixowie i Unixopodobne (GNU/Linux). S18T zostało napisane z wykorzystaniem języka Ruby.

Poniżej zawarto opis podstawowych klas zawartych w S18T. Opisywane metody zawierają informacje takie jak zwracana wartość ( wypisane są one kolorem **niebieskim** przed nazwą metody ), oraz w nawiasie wypisane jakie parametry metoda przyjmuje ( nazwa parametru jest

napisana kolorem czarnym. W nawiasie, w kolorze **jasnoniebieskim**, napisany jest jaki typ powinna mieć zmienna,  wartość po znaku '=' oznacza domyślną wartość parametru).

# 2 Klasa S18T::Cursor

Klasa obsługuje kursor w konsoli

## Metody

- **nil self.SetPosition ( _x(integer), _y(integer)=0)** – ustawia pozycję kursora na współrzędne podane w parametrach (_x i _y).
- **nil chPosition ( _x(integer), y(integer)=0) –** zmienia pozycję kursora o wartości podane w parametrach (_x i _y).
- **Array[] self.GetPosition ()** - zwraca w postaci tablicy, aktualną pozycję kursora.
- **nil self.GetColor ( _fontColor(integer)=9, _backgroudColor(integer)=9 )** - Ustawia kolor pisanego tekstu. Kolor czcionki i tła zależy od wartości podanych w parametrach (_fontColor i _backgroundColor).

# 3 Klasa S18T::Engine

Klasa zawiera metody do zarządzania silnikiem gry.

## Zmienne klasowe

- **@@scene (S18T::Scene)** – Przechowuj obiekt klasy Scena. Na tym obiekcie operuje silnik gry.
- **@@working (boolean)** – przechowuje informację czy silnik „pracuje".
- **@@lock (boolean)** – przechowuje informację czy wykonywana jest metoda S18T::Engine.MainLoop().
- **@@deltaTime (float)**– przechowuje czas męczy wywoływaniami metody MainLoop()
- **@@time (float)**– przechowuje czas systemowy jaki był w momencie wywołania metody S18T::Engine.MainLoop().
- **@@lastTime (float) –** przechowuje czas systemowy jaki był momencie przedostatniego wywołania metody S18T::Engine.MainLoop().

## Metody

- **nil self.Start ()** - inicjuje silnik. Wiąże się to ze zmianą działania terminalu ( ram mod; no echo ). Metoda ustawia zmienną @@working na true.
- **nil self.Stop ()** - zakończa pracę silnika i przywraca normalny tryb pracy terminalu. Metoda zmienna wartość zmiennej @@working na false.

- **nil self.SetScene ( _scene(S18T::Scene) )** - metoda służy do ustawienia danej sceny do obsługi przez silnik. Metoda przyjmuje jedne argument jakim powinien być obiekt klasy Scena. Metoda zmienia scenę natychmiast o ile nie wykonywana jest główna pętla. W innym przypadku, podana scena zostanie ustawiona po ukończeniu głównej pętli ( metody S18T::Engine.MainLoop() ). Po ustawieniu nowej sceny wywoływana jest metoda init danego obiektu sceny ( S18T::Scene.init() ).
- **nil self.MainLoop ()** - jej wywołanie wykonuje jedną „klatkę" gry i aktualizuje obraz gry. Metoda po pojedynczym wywołaniu wykonuje następujące czynności w danej kolejności:
  - aktualizuje czas silnika i wylicza różnice czasów względem ostatniego wywołania funkcji.
  - Odczytuje bufor standardowego wejścia (STDIN)
  - Wywołuje metodę update dla aktualnie obsługiwanego obiektu Sceny ( S18T::Scene.update() )
  - Wywołuje metodę obsługi obiektów z aktualnego obiektu Sceny przez klasę Screen ( S18T::Screen.ServiceObjects( S18T::Scene ) ).
  - Wywołuje metodę Draw klasy Screen do narysowana wyrenderowanej klatki.
  - Wywołuje metodę killDying() dla aktualnego obiektu Sceny by usunąć „umierające" obiekty gry.
  - Wywołuje metodę setSesvisedObjects() dla aktualnego obiektu Sceny by zaktualizować flagi obiektów gry.

  Na czas wykonywania powyższych czynności metoda ustawia zmienną @@lock na „true". Jest to zabezpieczeniem przed np. zmianą obiektu sceny podczas obsługi starej sceny. Opcjonalnie, na koniec metoda ustawia nowy obiekt Sceny.
- **integer self.GetDeltaTime ()** - zwraca różnicę czasu między wywołaniem metody MainLoop().
- **integer self.GetTime ()** - zwraca czas systemu który był w momencie ostatniego wywołania metody MainLoop().
- **S18T::Scene self.GetScene ()** - zwraca aktualny obiekt klasy Scena.
- **boolean self.Working? ()** - Funkcja zwraca wartość zmiennej @@working.

# 4 Klasa S18T::Frame

Klasa przechowuje klatkę gry. Klasa przy renderowaniu klatki tworzy „maskę" która informuje które piksele zostały zmienione i należy je wypisać do standardowego wyjścia (STDOUT).

## Zmienne

- **@size** – zawiera wymiary renderowanej klatki

- **@buffor** – przechowuje wyrenderowaną klatkę
- **@mask** – przechowuje informację o zmienionych pikselach
- **@changed** – przechowuje informację czy klatka została zmieniona.

## Metody

- **nil initialize ( _size(S18T::Vector2) )** - tworzy tablice o wymiarach zależnych od wartości obiektu Vecora2 podanego w parametrze.
- **nil clearField ( _position(S18T::Vector2), _sprite(S18T::Sprite) )** - czyści piksele klatki. Na podstawie pierwszego parametru (_position) określona zostaje pozycją czyszczonego obszaru, zaś z podanego sprite'a (_sprite) pobierane są wymiary czyszczonego obszaru. Wyczyszczone piksele zostają zastąpione nil'em.
- **nil paintDraw ( _position(S18T::Vector2), _sprite(S18T::Sprite) )** - rysuje podanego w parametrze sprite'a (_sprite) na klatce. Pierwszy parametr (_position) określa pozycję rysowanego sprite'a. Metoda ustawia wartość zmiennej @changed na true.
- **nil paintPixel ( _x(integer), _y(integer), _pixel(S18T::Pixel)=nil)** - rysuje pojedynczy piksel o współrzędnych podanych w parametrach (_x i _y). Na ich podstawie wyliczany jest wskaźnik do jednowymiarowej tablicy @buffor gdzie zostaje umieszczony obiekty Pixel (_pixel).
- **boolean printPixel ( _x(integer), _y(integer) )** - Wypisuje do standardowego wyjścia (STDOUT) piksel o podanych współrzędnych w parametrach (_x i _y). Metoda zmienia wartość maski danego piksela na 0. Jeśli piksel o danych współrzędnych jest pusty, metoda zwraca false, w innym wypadku true.
- **boolean isNil? ( _x(integer), _y(integer) )** - jeśli piksel o danych w parametrach współrzędnych (_x i _y) jest pusty, metoda zwraca false, w innym wypadku true.
- **boolean wasCh? ( _x(integer), _y(integer) )** - jeśli maksa piksela o danych w parametrach współrzędnych (_x i _y) jest równa 1, metoda zwraca true, w innym wypadku zwraca false.
- **boolean changed? ()** - zwraca wartość zmiennej @changed
- **Array[] getMask ()** - zwraca zmienną @mask
- **integer getBufforOffset ( _x(integer), _y(integer) )** - na podstawie współrzędnych podanych w parametrach (_x i _y ) wylicza pozycję w jednowymiarowej tablicy.
- **nil clear ()** - metoda czyści klatkę (@buffor) wypełniając ją nil'ami oraz maskę (@mask) wypełniająć ją zerami.
- **nil clearMask ()** - metoda czyści maskę (@mask) wypełniając ją zerami

# 5 Klasa S18T::Input

Obsługuje standardowe wejście (STDIN).

## Zmienne klasowe

- **@@chars –** tablica (Array) przechowująca odczytane znaki ze standardowego wejścia (STDIN).
- **@@inputStream** – przechowuje ciąg znaków odczytany ze standardowego wejścia (STDIN)
- **@@inited** – przechowuje informację czy zainicjowano klasę Input.

## Metody

- **nil self.Init ()** - Inicjuje klasę Input zmieniając tym obsługi standardowego wejścia (STDIN ) na RAW mode i no echo. Metoda zmienia zmienną @@inited na true.
- **nil self.Close ()** - przywraca normalny tryb pracy terminala i zmienna wartość zmiennej @@inited na wartość false.
- **nil self.Update ()** - aktualizuje tablicę odczytanych znaków ( @@chars )ze standardowego wejścia (STDIN) i strumień wejścia (@@inputStream). Jeśli klasa Input nie została zainicjowana (@@inited == false ) metoda przy wywołaniu wypisze komunikat o błędzie i zwróci nil.
- **nil self.Clear ()** - czyści zmienną @@chars i @@inited.
- **boolean self.Key ( _char( String or char ) )** - jeśli znak _char znajduje się w tablicy @@chars, metoda zwróci prawdę. W innym wypadku zwróci nil.
- **boolean self.KeyUD ( _char( String or char ) )** - działa podobnie jak metoda S18T::Input.Key( _char ); z tą różnicą że wielkość znaków nie ma znaczenia.
- **String self.getStream ()** - zwraca wartość zmiennej @@streamInput
- **integer self.getStatus ()** - Zwraca ilość odczytanych znaków ze standardowego wejścia (STDIN)
- **Array[] self.getKeys ()** - Zwraca tablicę @@chars w postaci stringu.
- **nil self.Error_noInit ()** - wypisuje informację o niezainicjowaniu klasy Input.
-

# 6 Klasa S18T::Object

Jest to klasa bazowa dla obiektów gry. Przechowywane są w obiektach klasy S18T::Scene. Wszystkie klasy, pełniące obiekty z gry muszą dziedziczyć po tej klasie.

## Zmienne

- **@name(String)** – nazwa obiektu
- **@position(S18T::Vector2)** – pozycja obiektu. Wartość należy zmieniać tylko przez metodę „setPosition” i „changePosition”.

- **@oldPosition(S18T::Vector2)** – stara pozycja obiektu, zmienna wykorzystywana przez klasę S18T::Screen do czyszczenia „zwolnionych” pikselów.
- **@changed(boolean)** – określa czy obiekt zmienił pozycję, co wiąże się ze aktualizacją klatki (S18T::Frame)
- **@created(boolean)** – określa że obiekt został dopiero stworzony i należy wywołać w nim metodę S18T::Object.start().
- **@dying(boolean)** – określa że obiekt jest „umierający” i należy wywołać w nim metodę S18T::Object.dead().
- **@sprite(S18T::Sprite)** – przechowuje obiekt klasy S18T::Sprite.
- **@zIndex(integer)** – definiuje pozycję „Z” (obecnie niewykorzystywane ).

## Metody

- **nil initialize( _args(Array[])=nil )** - na podstawie tablicy podanej w parametrze ustawiane są zmienne nowego obiektu. Parametry przekazywane przez tablice opisane są poniżej:
  - :name(String) – określa nazwę obiektu
  - :sprite(S18T::Sprite) – przekazuje sprite’a ( obiekt klasy S18T::Sprite).
  - :position(S18T::Vector2) – przekazuje pozycję obiektu
  - :zIndex(integer) – przekazuje wartość zIndex.
- **boolean eqName? ( _string(String) )** - porównuje nazwę obiektu (@name) z ciągiem znaków podanym w parametrze ( _string ). Jeśli są sobie równe, metoda zwraca true, w innym przypadku false.
- **boolean changed? ()** - zwraca wartość zmiennej @changed
- **boolean created? ()** - zwraca wartość zmiennej @created
- **boolean dying? ()** - zwraca wartość zmiennej @dying
- **nil setPosition ( _position(S18T::Vector2) )** - ustawia pozycję obiektu (@position) na tą podaną w parametrze (_position). Metoda również ustawia wartość zmiennej @changed na true, co jest istotne dla prawidłowego zaktualizowania klatki gry (S18T::Frame).
- **nil changePosition ( _position(S18T::Vector2) )** - działa podobnie jak S18T::Object.setPosition() z tą różnicą że przesuwa pozycję obiektu o współrzędne podane w parametrze (_position).
- **nil setServised ()** - Ustawia wartości @changed i @created na false. Metody powinna być wywołana jedynie przez obiekt klasy S18T::Screne.
- **nil setDying ()** - Ustawia wartość zmiennej @dying na true. Oznacza to że gdy Scena będzie obsługiwać ten obiekt, zostanie wywołana w nim metoda S18T::Object.dying() a następnie obiekt zostanie usunięty.
- **S18T::Vector2 getPosition ()** - zwraca pozycję obiektu (@position).

- **S18T::Sprite getSprite ()** - zwraca obiekt S18T::Sprite (@sprite).
- **integer getZIndex ()** - zwraca wartość zmiennej @zIndex.
- **nil start ()** - pusta metoda, może być nadpisana przez klasę dziedziczącą. Wywoływana jest przez obiekt klasy S18T::Scene przy pierwszym aktualizowaniu obiektu. Metodę można napisać przez dziedziczące klasy.
- **nil update ()** - pusta metoda, może być nadpisana przez klasę dziedziczącą. Wywoływana jest przy każdym aktualizowaniu sceny. Metodę można nadpisać przez dziedziczące klasy.
- **nil dead ()** - pusta metoda, może być nadpisana przez klasę dziedziczącą. Wywoływana jest przed usunięciem obiektu.

# 7 Klasa S18T::Pixel

Klasa przechowuje informację (znak oraz kolory) jednego piksela. Wykorzystywana jest przez klasy S18T::Sprite oraz S18T::Frame.

## Zmienne

- @char – znak piksela
- @color – kolor czcionki oraz tła piksela

## Metody:

- **nil initialize( _char(String lub char), _fontColor(integer)=9, _backgroudColor(integer)=9)** – tworzy piksel, jego zmienne ustawiane są na podstawie podanych parametrów.
- **String puts()** - zwraca w postaci „Escape code” kolory oraz znak piksela.

# 8 Klasa S18T::Scene

Klasa przechowuje obiekty gry. Odpowiada za ich aktualizację i usuwanie „umierających” obiektów.

## Zmienne

- **@objects(Array[])** - przechowuje obiekty gry

## Metody

- **nil initialize ()**
- **nil addObject ( _object(S18T::Object) )** - dodaje obiekt do tablicy @objects.
- **nil setServisedObjects ()** - wywołuje metodę S18T::Object.setServiced() dla wszystkich obiektów gry (S18T::Object) w tablicy @objects.

- **integer getNumberObjects ()** - zwraca liczbę obiektów w tablicy @objects.
- **S18T::Object getObject( _index(integer) )** - zwraca obiekt gry z tablicy @objects o indeksie podanym w parametrze (_index).
- **nil update ()** - aktualizuje wszystkie obiekty wywołując w nich metodę S18T::Object.update(). Dla nowo utworzonych obiektów ( ze zmienną @created == true ), wywołuje dodatkowo metodę S18T::Object.start().
- **nil killDying ()** - dla obiektów „umierających” ( ze zmienną @dying == true ), zostaje wywołana metoda S18T::Object.dead() a następnie zostają usunięte.
- **Array[] findObjectsByName ( _name(String) )** - zwraca tablicę obiektów gry (S18T::Object) ze zmienną @name równe ciągowi znaków podanemu w parametrze (_name).
- **nil killAll ()** - usuwa wszystkie obiekty gry (S18T::Object) z tablicy @objects, bez wywoływania metody S18T::Object.dead().
- **nil init ()** - pusta metoda, klasa dziedzicząca może ją nadpisać. Wywoływana jest przez S18T::Engine po ustawieniu sceny na aktualną.
- **nil close ()** - pusta metoda, klasa dziedzicząca może ją nadpisać. Przy zmianie sceny, klasa S18T::Engine wywołuje tą metodę.

# 9 Klasa S18T::Screen

Klasa odpowiada za renderowanie klatek i wypisywanie ich do standardowego wyjścia (STDOUT)

## Zmienne klasowe

- **@@winsize** – określa wielkość renderowanych klatek
- **@@fremes** – przechowuje obiekty klatek ( obecnie tylko jednej)
- **@@nullPixel** – definiuje co ma zostać wypisane do standardowego wyjścia (STDOUT) w miejsce pustego piksela.

## Metody

- **nil Init ( winsize(S18T::Vector2)=nil )** - inicjuje klasę tworząc  obiekt klasy Frame. Jeśli do metody podany zostanie obiekt klasy Vector2, wymiary renderowanych klatek zostaną ustawione względem parametrów wectora.
- **Vector2 GetTerminalSize ()** - zwraca klona obiektu Vector2 ze zmiennej @@winsize
- **nil ServiceObjects ( _scene(S18T::Vector2) )** - metoda renderuje klatkę na podstawie obiektów gry zawartych w podanej w parametrze obiekcie Sceny (S18T::Scene).
- **nil Draw ()** - metoda wypisuje do standardowego wyjścia wyrenderowaną klatkę.
- **nil NullPixel ()** - wypisuje pusty piksel (@@nullPixel)

- **nil Clear ()** - czyści terminal
- **Array[] CursorPositionFromIt ( it(integer) )** - na podstawie otrzymanej liczby (wskaźnika pikselu w tablicy jednowymiarowej ), zwraca współrzędne piksela.

# 10 Klasa S18T::Sprite

Przechowuje sprite'a obiektu w postaci tablicy obiektów klasy S18T::Pixel.

## Zmienne

- **@offset(S18T::Vector2) –** wskazuje punkt (0,0) danego sprite'a.
- **@size(S18T::Vector2) –** przechowuje wymiary sprite'a.
- **@draw(Array[])** - przechowuje obiekty klasy S18T::Pixel.

## Metody

- **nil initialize ( _size(S18T::Vector2)=S18T::Vector2.new(1,1), _offset(S18T::Vector2)=S18T::Vector2.new(0,0) )** - inicjuje obiekt ustawiając parametry @size na _size, @offset na _offset i przypisuje zmiennej @draw tablicę o wymiarach S18T::Vector2 ze zmiennej @size.
- **nil setPixel ( _x(integer), _y(integer), _pixel(S18T::Pixel) )** - przypisuje danej komórce z tablicy @draw, piksel podany w parametrze _pixel.
- **nil getOffset ()** - zwraca wartość zmiennej @offset.
- **nil getSize ()** - zwraca wartość zmiennej @size.
- **S18T::Vector2 getPixel ( _x(integer), _y(integer) )** - zwraca wartość komórki z tablicy @draw, o indeksie wyliczonym ze współrzędnych ( _x i _y ).

# 11 Klasa S18T::Vector2

Klasa ma za zadanie ułatwić operację na punktach,wektorach i wartościach dwuargumentowych.

## Zmienne

- **accesor @x(integet)** – przechowuje wartość x.
- **accesor @y(integer)** – przechowuje wartość y.

## Metody

- **nil initialize( _x(float), _y(float) )** - ustawia wartości zmiennych @x i @y odpowiednio na wartości parametrów _x i _y.
- **float value ()** - zwraca wartość wektora.
- **Array[] values ()** - zwraca wartość @x i @y.

- **S18T::Vector2 clone ()** - zwraca klona obiektu klasy S18T::Vector2, o identycznych wartościach @x i @y.
- **String to_s ()** - zwraca wartości @x i @y w postaci ciągu znaków.
- **float field ()** - zwraca pole prostokąta o wymiarach @x i @y.
- **boolean eql? ( _vector2(S18T::Vector2) )** - porównuje wartości @x i @y z wektorem otrzymanym w parametrze, jeśli są równe zwraca true, w innym wypadku false.
- **S18T::Vector2 + ( _vector2(S18T::Vector2) )** - do wartości @x i @y dodaje wartości z wektora podanego w parametrze (_vector2).
- **S18T::Vector2 - ( _vector2(S18T::Vector2) )** - od wartości @x i @y odejmuje wartości z wektora podanego w parametrze (_vector2).
- **S18T::Vector2 * ( _vector2(S18T::Vector2) )** -wartości @x i @y mnoży o wartości z wektora podanego w parametrze (_vector2).
- **S18T::Vector2 / ( _vector2(S18T::Vector2) )** -wartości @x i @y dzieli przez wartości z wektora podanego w parametrze (_vector2).
- **boolean == ( _vector2(S18T::Vector2) )** - jeśli wartość wektora jest równa wartości wektora podanego w parametrze (_vector2), zwraca true, w innym przypadku false.