Dokumentacja projektu WallpaperWeb

Release 1.0.0

Agnieszka Głowacka
Anastasiya Dorosh
Martyna Trębacz
Anna Waleczek
Oliwia Skucha
Jakub Rogoża
Krzysztof Emerling
Szymon Duda

Dec 27, 2025

CONTENTS:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CHAPTER

ONE

PYTHON FUNCTIONS

app.**get_locale**()

Pobiera preferowany język użytkownika zapisany w sesji.

Returns:

str: Kod języka (np. ‘en’ dla angielskiego), domyślnie ‘en’.

CHAPTER

TWO

JAVASCRIPT FUNCTIONS

`updateStats()`

Funkcja wyliczająca FPS i Frametime na podstawie różnicy wydajności w poprzedniej i aktualnej klatce.

`resizeCanvas()`

Funkcja obsługująca automatyczne dostosowywanie rozmiaru canvasu do aktualnego rozmiaru okna przegladarki.

`createShader`(*gl*, *type*, *source*)

Tworzy i kompiluje shader WebGL.

Arguments

- `gl` (**WebGL2RenderingContext**) – Aktywny kontekst WebGL2.
- `type` (**number**) – Typ shadera (*gl.VERTEX_SHADER* lub *gl.FRAGMENT_SHADER*).
- `source` (**string**) – Kod źródłowy shadera w języku GLSL.

Returns

WebGLShader|null – - Skompilowany shader lub *null* w przypadku błędu kompilacji.

`createProgram`(*gl*, *vertexShader*, *fragmentShader*)

Tworzy i linkuje program WebGL z shaderów wierzchołków i fragmentów.

Arguments

- `gl` (**WebGL2RenderingContext**) – Aktywny kontekst WebGL2.
- `vertexShader` (**WebGLShader**) – Skompilowany shader wierzchołków.
- `fragmentShader` (**WebGLShader**) – Skompilowany shader fragmentów.

Returns

WebGLProgram|null – - Zlinkowany program WebGL lub *null* w przypadku błędu linkowania.

`loadShaderSource`(*name*)

Asynchronicznie ładuje kod źródłowy shadera z pliku.

Arguments

- `name` (**string**) – Nazwa pliku shadera znajdującego się w katalogu */static/shaders/*.

Throws

`Error` –

- Gdy plik shadera nie może zostać załadowany.

Returns

Promise.<string> – - Kod źródłowy shadera w formacie tekstowym.

`init()`

Asynchronicznie ładuje wszystkie shadery wykorzystywane w aplikacji.

Returns
Promise.<{vertexShaderSource: string, fragmentShaderSource: string, fragmentAsciiShaderSource: string, fragmentFXAASource: string}> – - Obiekt zawierający kody źródłowe wszystkich shaderów.

`createTextureFromImage(`*gl*, *program*, *image*, *textureSlot*, *uniformName*`)`

Funkcja tworząca teksturę 2D z wczytanego obrazu.

Arguments

- **gl** (**WebGL2RenderingContext**) – wskaźnik na kontekst gl.
- **program** (**WebGLProgram**) – wskaźnik na program.
- **image** (**TexImageSource**) – plik obrazu.
- **textureSlot** (**number**) – wskaźnik na slot w który ładujemy teksturę.
- **uniformName** (**string**) – nazwa uniformu pod który podpinamy teksturę.

Returns
WebGLTexture – - obiekt tekstury

`createRenderTarget(`*gl*, *width*, *height*`)`

Tworzy bufor ramki (Framebuffer) z teksturą jako celem renderowania.

Arguments

- **gl** (**WebGL2RenderingContext**) – Aktywny kontekst WebGL2.
- **width** (**number**) – Szerokość tekstury render targetu w pikselach.
- **height** (**number**) – Wysokość tekstury render targetu w pikselach.

Returns
Object – - Obiekt zawierający framebuffer oraz powiązaną z nim teksturę.

`toggleScene()`

Funkcja obsługująca zmiane sceny.

`vector1i(`*x_*`)`

Funkcja tworząca jednowymiarowy wektor INT.

Arguments

- **x_** (**int**) – wartość x

Returns
wektor jednowymiarowy w formie listy [x]

`vector2i(`*x_*, *y_*`)`

Funkcja tworząca dwuwymiarowy wektor INT.

Arguments

- **x_** (**int**) – wartość x
- **y_** (**int**) – wartość y

Returns
wektor dwuwymiarowy w formie listy [x, y]

`vector3i`(*x_*, *y_*, *z_*)

Funkcja tworząca trójwymiarowy wektor INT.

Arguments

- **x_** (**int**) – wartość x
- **y_** (**int**) – wartość y
- **z_** (**int**) – wartość z

Returns

wektor trójwymiarowy w formie listy [x, y, z]

`vector4i`(*x_*, *y_*, *z_*, *t_*)

Funkcja tworząca czterowymiarowy wektor INT.

Arguments

- **x_** (**int**) – wartość x
- **y_** (**int**) – wartość y
- **z_** (**int**) – wartość z
- **t_** (**int**) – wartość t

Returns

wektor czterowymiarowy w formie listy [x, y, z, t]

`vector1f`(*x_*)

Funkcja tworząca jednowymiarowy wektor FLOAT.

Arguments

- **x_** (**float**) – wartość x

Returns

wektor jednowymiarowy w formie listy [x]

`vector2f`(*x_*, *y_*)

Funkcja tworząca dwuwymiarowy wektor FLOAT.

Arguments

- **x_** (**float**) – wartość x
- **y_** (**float**) – wartość y

Returns

wektor dwuwymiarowy w formie listy [x, y]

`vector3f`(*x_*, *y_*, *z_*)

Funkcja tworząca trójwymiarowy wektor FLOAT.

Arguments

- **x_** (**float**) – wartość x
- **y_** (**float**) – wartość y
- **z_** (**float**) – wartość z

Returns

wektor trójwymiarowy w formie listy [x, y, z]

`vector4f`(*x_*, *y_*, *z_*, *t_*)

Funkcja tworząca czterowymiarowy wektor FLOAT.

Arguments

- **x_** (**float**) – wartość x
- **y_** (**float**) – wartość y
- **z_** (**float**) – wartość z
- **t_** (**float**) – wartość t

Returns

wektor czterowymiarowy w formie listy [x, y, z, t]

`normalize`(*array*)

Funkcja normalizująca każdą wartość w liście do zakresu [0, 1].

Arguments

- **array** (**list**) – tablica

Returns

znormalizowana tablica.

`starsGenerator`(*seed*, *minDistance*, *K*)

Funkcja obsługująca pobieranie wartości do generatora gwiazd.

Arguments

- **seed** (**int**) – ziarno
- **minDistance** (**int**) – minimalna odleglość pomiędzy dwoma punktami
- **K** (**int**) – ilość prób podjęta do znalezienia pasującego punktu

Returns

wypłaszczony wektor zawierający współrzędne gwiazd [x1, y1, x2, y2, . . . , xn, yn].

`gaussian`(*x*, *sigma*)

Funkcja pomocnicza do wyznaczania wartości jednowymiarowej funkcji Gaussa.

Arguments

- **x** (**int**) – wartość
- **sigma** (**float**) – rozmycie sigma

Returns

wartość funkcji Gaussa.

`gaussianBlur`(*kernelSize_handler*, *intensity_handler*)

Funkcja obsługująca pobieranie wartości do shadera Gaussian Blur.

Arguments

- **kernelSize_handler** (**int**) – wielkość kernela
- **intensity_handler** (**float**) – intensywność efektu

Returns

gotowe wartości dla shadera.

bloom(*bloomIntensity_handler*, *bloomKernelSize_handler*)

Funkcja obsługująca pobieranie wartości do shadera Bloom.

Arguments

- **bloomIntensity_handler** (**float**) – intensywność efektu
- **bloomKernelSize_handler** (**int**) – wielkość kernela

Returns

gotowe wartości dla shadera.

isChecked(*element*)

Sprawdza czy przycisk jest wciśnięty.

Arguments

- **element** (**object**) – obiekt DOM

Returns

boolean true/false.

sliderValue(*slider*, *input*)

Funkcja pilnująca by wprowadzana wartość nie wchodziła poza zakres <min, max>.

Arguments

- **slider** (**object**) –
  - input type Range
- **input** (**object**) – input type Number

restoreDefault(*input*)

Funkcja ustawiająca wartość na minimum jeżeli zostanie całkowicie usunięta z text area.

Arguments

- **input** (**object**) – input type Number

inputValue(*slider*, *input*)

Funkcja ustawiający input.value takie samo jak w sliderze.

Arguments

- **slider** (**object**) – input type Range
- **input** (**object**) – input type Number

inputValidation(*input*)

Funkcja pilnująca żeby w inpucie nie można było przekroczyć wartości minimalnej i maksymalnej.

Arguments

- **input** (**object**) – input type Number

saveSessionData()

Funkcja zapisująca dane sesji.

fetchSceneValues()

Funkcja pobierająca aktualne wartości w danej scenie.

Returns

array

updateSceneValues(*array*, *scene*)

Funkcja aktualizująca wartości w tablicy dla podanej sceny.

Arguments

- **array** (**dict**) – tablica z wartościami
- **scene** (**string**) – wskaźnik na aktywną scene

setSceneValues(*array*, *scene*)

Funkcja ustawiająca wartości dla podanej sceny.

Arguments

- **array** (**dict**) – tablica z wartościami
- **scene** (**string**) – wskaźnik na aktywną scene

updateSceneShaders(*scene1*, *scene2*)

Ukrywa shadery niedostępne w wybranej scenie.

Arguments

- **scene1** (**list**) – tablica zawierająca liste shaderów dostępnych na scenie 1
- **scene2** (**list**) – tablica zawierająca liste shaderów dostępnych na scenie 2

hideButton()

Ukrywa przycisk do renderowania sceny 1

createJSON()

Tworzy plik JSON i zapisuje do niego ustawienia aplikacji.

loadJSON()

Funkcja wczytuje plik JSON i odczytuje zapisane w nim ustawienia aplikacji.

Returns

sparsowany plik JSON.

INDEX