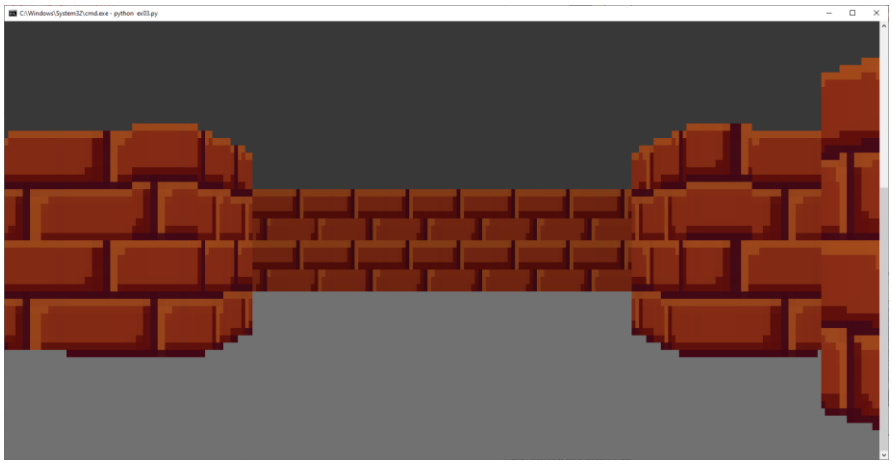
Cílem třetí úlohy je vytvoření rozšiřujícího modulu začlenitelného do implementace základu jednoduché FPS hry z předchozího zadání. Účelem modulu je nahradit původně jednobarevné zdi jejich otexturovanou verzí a tím dále zlepšit dojem z hloubky vykreslované scény. Očekávaný výsledek by pro texturu cihlové zdi mohl vypadat takto (pozn., samotné propojení výsledků úloh č. 2 a 3 bude předmětem dalšího úkolu):



Textura bude načtena ze souboru uloženého v jednoduchém obrazovém formátu PPM [1, 2]. Následující body by vás měly postupně provést detaily implementace tohoto úkolu.

Úloha 1: (1 bod)

Vytvoření třídy pro reprezentaci textury.

V samostatném souboru `texture.py` vytvořte třídu `Texture`, která bude potomkem třídy `object`. Konstruktor třídy bude přijímat textový řetězec `file_name` reprezentující název obrazového souboru ve formátu PPM. Třída bude dále obsahovat instanční atribut `data` typu `list` pro uložení načtených dat. Jednotlivé texely (obrazový element textury) budou reprezentovány jako trojice ve formátu (R, G, B), kde každý kanál bude typu float. Např. níže uvedená trojice bude v nelineárním (gamma komprimovaném) formátu sRGB odpovídat hnědé barvě:

(0.514, 0.165, 0.078) ⟺ 

Posledními dvěma instančními atributy naší nové třídy budou šířka (`width`) a výška (`height`) uložené textury. Oba atributy budou v konstruktoru inicializovány na nulu.

Úloha 2: (1 bod)

Načtení textury z PPM souboru.

Vytvořte metodu třídy `Texture` pojmenovanou `load` přijímající textový řetězec `file_name` stejně jako konstruktor. Tato metoda načte ze zadaného souboru texturu uloženou ve formátu P3 (ASCII Portable PixMap). Zkráceny výpis z dodaného testovacího souboru `wall.ppm` vypadá takto:

```
P3
# Texture of the seamless brick wall
16 16
255
177 80 31 177 80 30 177 80 30 177 81 31 171 77 31
77 12 25 177 80 30 178 82 31 177 81 31 177 80 30
…
75 9 24
```

Hlavička souboru se sestává z řetězce `'P3'`, což je dvoubajtové magické číslo udávající formát souboru, následuje nepovinný komentář uvozený znakem mřížka (`'#'`), další dvojice čísel představuje výšku a šířku textury a číslo 255 reprezentuje maximální hodnotu, která se může vyskytnout v datové části souboru. V datové části jsou pak postupně uloženy jednotlivé texely ve formátu RGB. Tyto hodnoty postupně načtěte, znormalizujte podle uvedené maximální hodnoty a uložte do atributu `data` jako seznam trojic. Dodejme, že matice texelů je organizována po řádcích. Zapamatujte si rovněž šířku a výšku textury v příslušných atributech třídy. Nakonec zavolejte metodu `load` přímo z konstruktoru třídy.

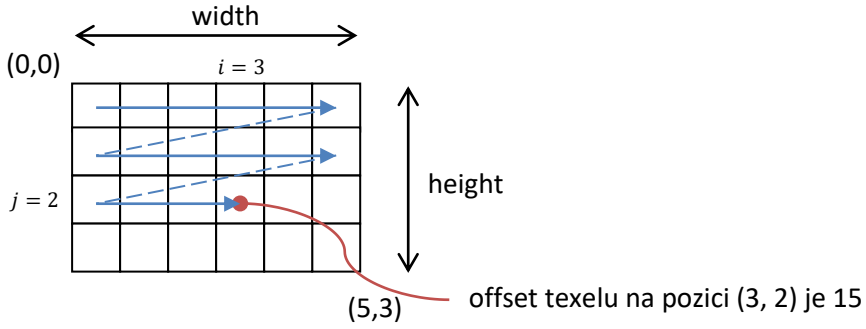
Po provedení metody `load` nad přiloženým souborem by měl seznam `data` vypadat následovně:

```
[(0.694, 0.314, 0.122), (0.694, 0.314, 0.118), …, (0.294, 0.035, 0.094)]
```

Úloha 3: (1 bod)

Pohyb po textuře.

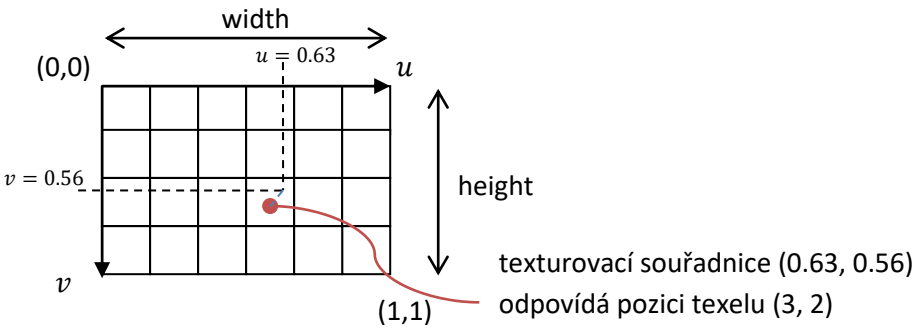
Vytvořte privátní metodu `offset`, která bude přijímat dva parametry `i`, `j` a vrátí offset texelu na pozici (i, j) viz obrázek:



Úloha 4: (1 bod)

Získání texelu z textury.

Třídu `Texture` rozšiřte o metodu `get_texel`, která bude přijímat souřadnice požadovaného texelu ve formě dvou reálných parametrů `u` a `v` z rozsahu $\langle 0,1 \rangle$. Výstupem metody bude texel (tj. trojice reálných čísel) odpovídající barvě textury na zadaných relativních (tzv. texturovacích) souřadnicích. Zde musíme použít interpolaci a pro jednoduchost se omezíme na tu nejjednodušší – interpolaci nejbližším sousedem. Parametry `u` a `v` převeďte na celočíselné indexy odpovídajícího texelu viz obrázek:



Z vypočtených indexů (i, j) pak s pomocí metody `offset` získejte pozici hledaného texelu v seznamu `data` a vraťte ho.

Reference

[1] http://netpbm.sourceforge.net/doc/ppm.html
[2] https://en.wikipedia.org/wiki/Netpbm