



Politechnika
Wroclawska

Grafika komputerowa i komunikacja człowiek-komputer

Laboratorium nr 6
Tekstutowanie obiektów

Szymon Datko
szymon.datko@pwr.edu.pl

Wydział Informatyki i Telekomunikacji,
Politechnika Wroclawska

semestr zimowy 2022/2023



Cel ćwiczenia

1. Zapoznać się i zrozumieć zagadnienie teksturowania.
2. Poznać sposób aplikowania tekstury na zdefiniowane powierzchnie.
3. Nauczyć się jak stworzyć własną teksturę dla własnego obiektu.

W jaki sposób narysować złożone scenerie?



Źródło obrazka:

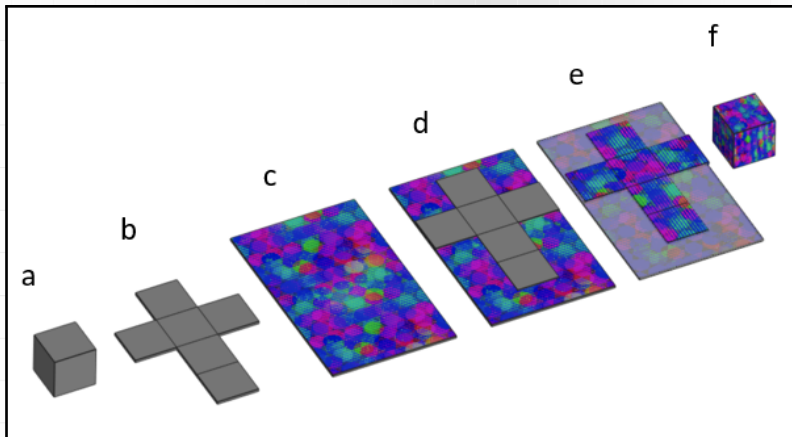
<https://magazyn.ceneo.pl/artykuly/Zagraj-bezplatnie-w-Wolfenstein-3D-za-pomoca-przegladarki>

Tekstury

- Bufory ze zdefiniowanymi mapami kolorów.
- Intuicyjnie: obrazki rozciągane pomiędzy wierzchołkami.
- Mogą stanowić dane wejściowe, jak i miejsce na zapis wyniku prac.
- Najczęściej wykorzystywane w shaderze fragmentów, poprzez:
 - obiekty typu `uniform sampler` (odniesienie do bufora),
 - określenie koloru fragmentu przez współrzędne w teksturze.
- Podstawowe rodzaje tekstur:
 - 1-wymiarowe, - 2-wymiarowe, - 3-wymiarowe.
- Kiedyś ograniczone: kwadratowe, rozmiary boków jako potęgi liczby 2,
 - wciąż jednak lepiej jest trzymać się tych założeń co do tekstur.

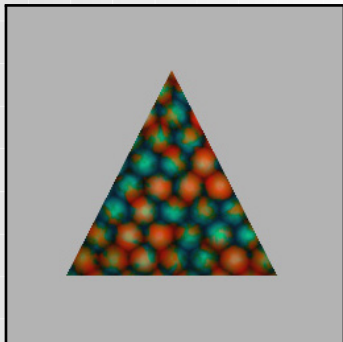
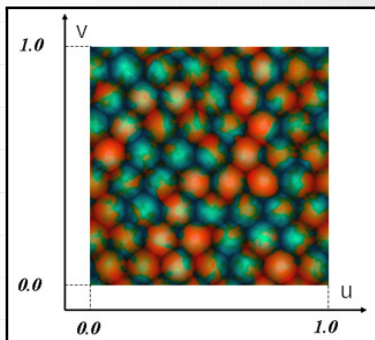
Teksturowanie – idea

- ▶ Do wierzchołków modelu mapuje się punkty z przestrzeni UV tekstury.
- ▶ W Legacy OpenGL stosuje się do tego funkcję `glTexCoord()`.

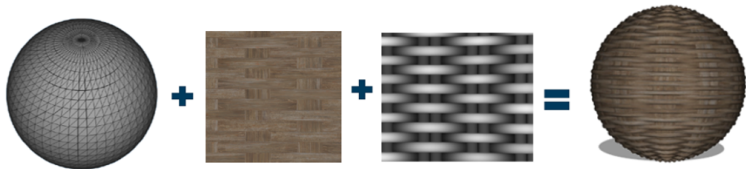


Tekstura w kontekście OpenGL

- ▶ Po wczytaniu, tekstura zostaje zmapowana do przestrzeni jednostkowej.
- ▶ Zastosowane tu współrzędne tekstury: $(0.0, 0.0)$, $(1.0, 0.0)$, $(0.5, 1.0)$.
- ▶ Wywołanie `glTexCoord()` następuje tuż przed `glVertex()`.



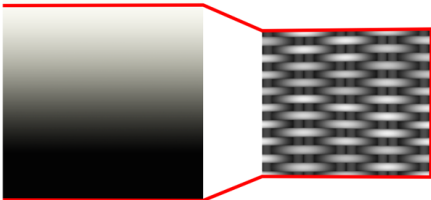
Tekstury to nie tylko mapy kolorów!



Mesh Geometry + Texture + Height Map = Full Part Realism

White = High

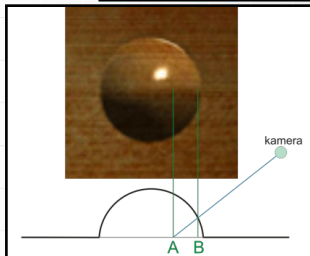
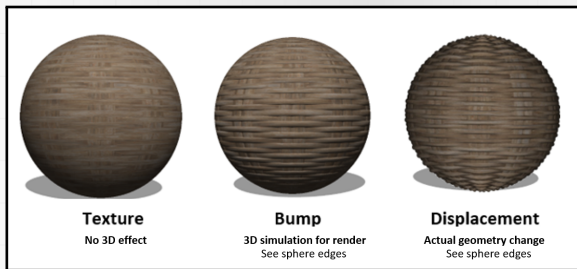
Black = Low



Źródło:

<https://grabcad.com/tutorials/texture-bump-and-displacement-how-to-make-photorealistic-models>

Praca na fragmentach, a praca na geometrii



Źródło: Dr inż. Rafał Wcisło "Rendering czasu rzeczywistego" (AGH) [Mapowanie paralaksy],

<https://grabcad.com/tutorials/texture-bump-and-displacement-how-to-make-photorealistic-models>

Zawijanie i filtrowanie tekstur

- ▶ Domyślnie każda tekstura, niezależnie od rozmiaru w pikselach, jest mapowana do układu współrzędnych jednostkowych.
- ▶ Wybrane wartości i zaokrąglenia mogą skutkować artefaktami.



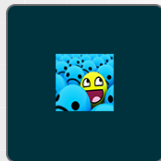
GL_REPEAT



GL_MIRRORED_REPEAT



GL_CLAMP_TO_EDGE



GL_CLAMP_TO_BORDER



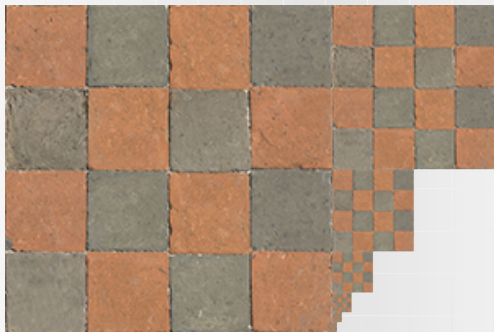
GL_NEAREST



GL_LINEAR

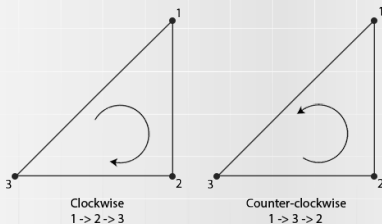
Mipmapy

- ▶ Nawet przy zastosowaniu filtrowania, mogą się pojawić problemy przy rysowaniu małych, daleko odległych obiektów.
- ▶ Problemy zaokrąglenia rozwiązuje użycie mniejszych tekstur.
- ▶ Mogą być wczytane, lub wygenerowane z oryginalnej tekstury.



Usuwanie niewidocznych ścian

- ▶ Zwykle rysowane są wszystkie powierzchnie w bryle widzenia.
- ▶ Jedna strona tych powierzchni najczęściej nie jest widoczna.
- ▶ Tzw. *Face Culling* pozwala zaoszczędzić część obliczeń GPU.
- ▶ Jest to też pomocne przy rysowaniu cienkich obiektów.
- ▶ Działanie mechanizmu (to, co widać) jest konfigurowalne.
- ▶ Wykorzystuje się tutaj właściwość iloczynu wektorowego.



Nowości w przykładowym programie (1/2)

- ▶ Załadowano moduł, pozwalający wczytywać obrazy w języku Python.

```
from PIL import Image
```

- ▶ Dodano funkcje związane z uaktywnieniem tekstuowania.

```
1| def startup():
2|     ...
3|
4|     glEnable(GL_TEXTURE_2D)
5|     glEnable(GL_CULL_FACE)
6|     glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE)
7|     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
8|     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
9|
10|    image = Image.open("tekstura.tga")
11|
12|    glTexImage2D(
13|        GL_TEXTURE_2D, 0, 3, image.size[0], image.size[1], 0,
14|        GL_RGB, GL_UNSIGNED_BYTE, image.tobytes("raw", "RGB", 0, -1)
15|    )
```

- Linie 4-8 włączają mechanizm tekstuowania i opisują jego ustawienia.
- Linie 10-15 to wczytanie obrazka z dysku i załadowanie go do pamięci.

Nowości w przykładowym programie (2/2)

- ▶ W funkcji `render()` narysowano trójkąt z mapowaniem tekstury.

```
1| def render(time):
2|     global theta
3|
4|     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
5|     glLoadIdentity()
6|
7|     gluLookAt(viewer[0], viewer[1], viewer[2],
8|               0.0, 0.0, 0.0, 0.0, 1.0, 0.0)
9|
10|    if left_mouse_button_pressed:
11|        theta += delta_x * pix2angle
12|
13|    glRotatef(theta, 0.0, 1.0, 0.0)
14|
15|    glBegin(GL_TRIANGLES)
16|    glTexCoord2f(0.0, 0.0)
17|    glVertex3f(-5.0, -5.0, 0.0)
18|    glTexCoord2f(1.0, 0.0)
19|    glVertex3f(5.0, -5.0, 0.0)
20|    glTexCoord2f(0.5, 1.0)
21|    glVertex3f(0.0, 5.0, 0.0)
22|    glEnd()
23|
24|    glFlush()
```

Koniec wprowadzenia.

Zadania do wykonania...

Zadania do wykonania (1)

Na ocenę **3.0** należy narysować oteksturowany kwadrat.

Wskazówki:

- kwadrat powinien być widoczny tylko z jednej strony,
 - ▶ chodzi o zobaczenie działania tak zwanego *face culling*,
 - ▶ jeśli występuje problem z widocznością, należy zmienić kolejność wierzchołków (czyli wywołań `glVertex()` i z nimi związanych),
- cały plik tekstury powinien zmieścić się na powierzchni kwadratu,
 - ▶ bez żadnych widocznych artefaktów łączenia, itd.
- zainstalować bibliotekę *Python Imaging Library*, jeśli jej brakuje,
 - ▶ `pip3 install --user Pillow`

Zadania do wykonania (2)

(po zrealizowaniu zadania poprzedniego)

Na ocenę **3.5** należy przygotować oteksturowany ostrosłup.

Wskazówki:

- podstawą bryły powinien być kwadrat z poprzedniego zadania,
 - ▶ z boków kwadratu powinny wychodzić trójkąty do wspólnego wierzchołka,
 - ▶ ten dodatkowy wierzchołek powinien znaleźć się nad środkiem kwadratu,
- teksturę zmapować w taki sposób, aby cała ona znalazła się na wystającej stronie tworzonej bryły (przy patrzeniu “z góry” widoczna jako kwadrat),
 - ▶ każdy trójkąt powinien zawierać ćwiartkę wejściowego obrazu,
 - ▶ widocznym efektem będzie przestrzenny obraz z podstawy,
- należy umożliwić ukrywanie i pokazywanie przynajmniej jednej ze ścian,
 - ▶ obsłużyć to na przykład za pomocą zdarzenia naciśnięcia klawisza,
- *face culling* powinien pozostać aktywny w ramach tego zadania.

Zadania do wykonania (3)

(po zrealizowaniu zadania poprzedniego)

Na ocenę **4.0** należy stworzyć i zastosować własną teksturę.

Wskazówki:

- utworzyć plik graficzny o rozmiarze na przykład 256x256 pikseli,
- wykorzystać na przykład program **Gimp** (GNU Image Manipulation Program),
 - ▶ zapis pliku za pomocą Plik > Eksportuj jako...,
 - ▶ format TGA,
 - ▶ orientacja Bottom Left (Dolny Lewy),
 - ▶ eksport koniecznie BEZ kompresji,
- tekstura może być abstrakcyjna, albo przedstawiać wizerunek jakiegoś zwierzęcia, bohatera dzieł kultury lub postać znanego celebryty ;-)

Zadania do wykonania (4)

(po zrealizowaniu zadania poprzedniego)

Na ocenę **4.5** należy wprowadzić możliwość przełączania tekstur.

Wskazówki:

- końcowy efekt, jaki chcemy uzyskać, to
 - ▶ dwa niezależnie otekstutowane obiekty obok siebie,
 - ▶ lub przełączanie (np. klawiszem) tekstur na jednym obiekcie,
- pliki tekstur (obrazki) mogą być wczytane wcześniej w programie,
 - ▶ przechowywane w pamięci RAM, czyli w ramach zmiennych,
 - ▶ nie otwierać pliku na dysku i nie czytać go w funkcji `render()` (!),
- teksturę w pamięci karty graficznej można podmienić za pomocą wywołania `glTexImage2D()` ze wskazaniem nowych danych tekstury.

Zadania do wykonania (5)

(po zrealizowaniu zadania poprzedniego)

Na ocenę **5.0** należy nałożyć teksturę na jajko.

Wskazówki:

- współrzędne wierzchołków modelu są wyznaczone z przestrzeni jednostkowej (u, v) , która nieprzypadkowo jest również jednostkowa,
 - ▶ można to wykorzystać do zmapowania współrzędnych tekstury,
 - ▶ jedną całą teksturę należy nałożyć na cały obiekt (lub połówkę),
 - unikać powielania całej tekstury na każdym kwadracie siatki modelu,
- znając budowę naszego modelu, zasadniczo podejścia są dwa:
 - ▶ odpowiednio przeliczyć współrzędne tekstury na powierzchni jajka,
 - ▶ przygotować teksturę, pasującą do wykorzystania zmiennych u i v ,
- *face culling* powinien pozostać aktywny w ramach tego zadania,
 - ▶ na połówce konieczna może być zmiana kolejności wierzchołków.