**!!! Program może wywołać atak epilepsji u epileptyków przejawiających objawy epilepsji.**

STEREOGRAMY

Dokumentacja programu

*Maria Korkuć:*

*Budowa interfejsu,*

*implementacja wczytywania obrazu,*

*zapisu do pliku*

*Wojciech Korzybski:*

*Algorytm losujący i przesuwający piksele*

*Karol Kocierz:*

*Algorytm tworzący maskę,*

# Opis projektu

Projekt dotyczy techniki tworzenia stereogramów opartych na przypadkowym rozkładzie punktów.

# Założenia wstępne

Bazowe funkcjonalności projektu:

1. Generowanie stereogramu na podstawie bitmapy.
2. Ładowanie bitmapy z pliku .bmp.
3. Wyświetlanie w rozdzielczości 800x600 pikseli.
4. Zapis stereogramu do pliku.

Dodane funkcjonalności:

1. Wyświetlanie kropek pomocniczych.
2. Podświetlanie maski na jednym z paneli stereogramu.
3. Wprowadzenie wyboru rozdzielczości.
4. Możliwość edycji koloru stereogramu
5. Dodanie możliwości wyboru wartości odcięcia maski.

# Analiza projektu

Do programu można załadować mapę głębokościową w skali szarości w formacie BMP. Mapa powinna być skonstruowana w taki sposób, ażby im punkt jest bliżej tym jest jaśniejszy. W interfejsie jest możliwość wyboru:

* Rozdzielczości
* Koloru stereogramu
* Pomagaczy (kropki pomocniczej, podświetlenia maski)
* Wartości odcięcia maski

Program może zapisać wygenerowany stereogram do pliku BMP.

Program przechowuje wygenerowane obrazy jako zmienne typu wxImage, stworzoną maskę w tablicy typu int, wczytana mapa w zmiennej typu wxBitmap. Ponadto wszystkie funkcje wykorzystywane do tworzenia stereogramu przechowywane są w klasie Stereogram!

Interfejs graficzny programu:



Przycisk wywołuje okno pozwalające wczytać mapę wysokościową.



Przycisk generuje nowe losowe punkty do stereogramu.



Rubryka aktywuje punkty pomocnicze.



Rubryka podświetla maskę.



Wybór rozdzielczości stereogramu.



Suwak wybiera wartość odcięcia maski.



Przyciski otwierają okna wyboru kolorów dla kropek tworzących stereogram.



Przycisk otwiera okno umożliwiające zapis stereogramu do pliku.

Projekt składa się z aż 3 zasadnych modułów:

* Moduł interfejsu w ramach którego wchodzi obsługa przycisków.
* Moduł wczytująco-zapisujący który wczytuje mapę i zapisuje stereogram.
* Moduł generujący stereogram który generuje stereogram.

Projekt został wykonany za pomocą biblioteki wxWidgets. Biblioteka ta została wybrana ze względu na łatwość z jaką implementuje się w niej interfejs. Użyte środowisko to Visual studio 201x community wraz z wbudowanym kompilatorem wybrane z powodu przejrzystego debugera.

# Analiza czasowa

Projekt był wykonywany w 4 sesjach:

1. Zaprojektowanie i stworzenie interfejsu – 2 godziny.
2. Implementacja algorytmów zapisu, wczytywania oraz tworzenia maski– 3 godziny.
3. Implementacja algorytmów tworzenia stereogramów – 4 godziny.
4. Refaktoryzacja oraz testowanie – 6 godzin.

# Kodowanie

**Opis funkcji zawartych w projekcie z podziałem na klasy w których się zawierają:**

**Klasa Stereogram**

1. wxImage movePixels(int\* mask, wxImage rawDots, bool\* flags);

Metoda klasy Stereogram, przesuwająca piksele na jednej połowie stereogramu, zapewniając jego poprawne działanie. Funkcja, iterując po całej wysokości obrazu oraz po połowie szerokości, za pomocą tablicy typu *int* (argument *mask*) sprawdza czy dany piksel ma zostać przesunięty. Jeśli nie, piksel zostaje niezmieniony. Jeśli tak, ustawia RGB danego piksela na RGB piksela przesuniętego o 10 pikseli w lewo. Jeżeli flaga informująca o zaznaczeniu opcji *Highlight* jest podniesiona, zmienia składnik B o 128. Ostatecznie funkcja zwraca zmieniony obiekt *wxImage*.

2. wxImage createDots(wxColour bacgroundColour, wxColour dotsColour);

Metoda klasy Stereogram, tworząca obraz losowych kropek. Na początku funkcja uaktualnia rozmiar obrazu i alokuje pamięć na odpowiednią ilość pikseli. Następnie, iterując po całej wysokości oraz połowie szerokości losuje, czy piksel powinien być koloru tła czy koloru kropki, ustawiając tak samo położenie na obu częściach stereogramu. Następnie maluje piksele zgodnie z wybranymi na paletach kolorami (odpowiednio argumenty *backgroundColour* oraz *dotsColour*). Obraz zapisywany jest jako obiekt *wxImage* i zwracany.

3. Gettery i settery:

int getWidth(); - getter szerokości

int getHeight(); - getter wysokości

void setWidth(int newWidth); - setter szerokości

void setHeight(int newHeight); - setter wysokości

Klasa Stereograms\_Frame

1. void Random\_Dots(wxCommandEvent& event);

Metoda klasy Stereogram\_Frame, powiązana z przyciskiem *Random dots*, wyświetlająca obraz losowanych kropek korzystając z funkcji *Draw()*, ustawiając uprzednio odpowiednią flagę oznaczającą, że losowanie ma się odbyć na nowo.

2. void Load\_Bitmap(wxCommandEvent& event);

Metoda klasy Stereograms\_Frame, uruchamiająca się po wciśnięciu przycisku *Load*. Wczytuje bitmapę, rozpoczynając od wyświetlenia okna dialogowego pozwalającego wybrać plik o rozszerzeniu BMP, którego następnie przypisuje do składowej klasy *\_bitmap* o typie *wxBitmap*. Metoda ta również zaznacza odpowiednie flagi, odpowiadające za informację o obecności bitmapy, a także o potrzebie wykonania nowej maski. Na końcu metoda uaktualnia output za pomocą funkcji *Draw()*.

3. void Save\_File(wxCommandEvent& event);

Metoda klasy Stereograms\_Frame, zapisująca bitmapę ze stereogramem do pliku przy wciśnięciu przycisku *Save*. Otwiera okno dialogowe pozwalające zapisać plik pod wybraną nazwą, w formacie BMP. Zapisuje do zmiennej *wxMemoryDC* bitmapę utworzoną na podstawie aktualnego stanu składowej *m\_panel* (typu *wxScrolledWindow\**), dzięki czemu jest w stanie następnie zapisać plik pod odpowiednią ścieżką. Na końcu odświeża wyświetlacz za pomocą funkcji *Draw()*.

4. void resize(wxSizeEvent& event);

Metoda klasy Stereograms\_Frame, uaktualniająca za pomocą funkcji *Draw()* rozmiar wyświetlanego obrazu.

5. void Make\_brighter(wxCommandEvent& event);

Metoda klasy Stereograms\_Frame, odpowiadająca za działanie w przypadku zaznaczenia pudełka podpisanego *Highlight*. Podnosi flagę oznaczającą konieczność zmiany składowej B odpowiednich pikseli przy generowaniu losowych kropek, a następnie odświeża obraz korzystając z metody *Draw().*

6. void Display\_resolution(wxCommandEvent& event);

Ta metoda obsługuje menu wyboru rozdzielczości. Ustawia wartości \_*pixelWidth* i \_*pixelHeight* według wybranej opcji.

7. void OnScrollBar(wxScrollEvent& event);

Metoda aktualizuje maskę kiedy przesuwany jest suwak zmieniający wartość odcięcia maski.

8. void Background\_color(wxColourPickerEvent& event);

Metoda ustawia kolor tła wedle wyboru z pierwszego przycisku wyboru koloru. Ustawia flagę sygnalizującą nowe losowanie kropek.

9. void Dots\_color(wxColourPickerEvent& event);

Metoda ustawia kolor kropek wedle wyboru z drugiego przycisku wyboru koloru. Ustawia flagę sygnalizującą nowe losowanie kropek.

10. void Helpful\_dot(wxCommandEvent& event);

Metoda ustawia flagę sygnalizującą konieczność wyrysowania kropki pomocniczej i ją rysuje.

11. void Draw();

Metoda sprawdza wszystkie flagi i na ich podstawie tworzy ostateczny obraz i wyrysowuje go na panelu interface’u.

12. void \_loadMask(int threshold, wxBitmap\* bitmap);

Metoda generuje maskę na podstawie bitmapy bitmap ustawiając w tablicy *mask* 1 na miejscach korespondujących pikselom które na bitmapie mają jasność większą od wartości przesłanej w threshold. Reszta miejsc jest ustawiona na 0.

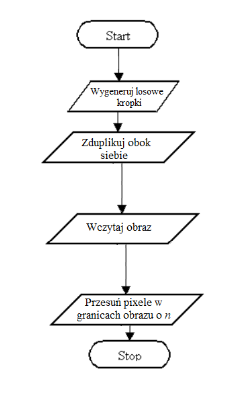
**Algorytm ogólny tworzenia stereogramów**

1. Wygeneruj losowe kropki na ekranie

2. Zduplikuj obraz stworzony z kropek i ustaw kopię obok oryginału

3. Wczytaj obraz

4. Przesuń piksele w granicach obrazu o n miejsc



# Testowanie

Ze względu na charakter projektu testowanie było utrudnione. Większość wyników zwracanych przez funkcje wymagało kontekstu oraz interpretacji zanim określone zostało czy wynik jest poprawny. Dlatego testowanie odbyło się na 2 różne sposoby:

1. Testowanie funkcji tworzących stereogram oraz losujących punkty. Wyłączając wszelkie pozostałe funkcje programu uruchamiano kilkukrotnie procedurę losującą sprawdzając za każdym razem czy nie są widoczne, żadne powtarzające się wzory. Następnie ustawiono na sztywno domyślną maskę w postaci prostokątu i sprawdzano czy powstanie odpowiedni stereogram.

Testowanie procedur zapisu oraz wczytywania. Na podstawie 2 obrazków BMP testowano poprawne wczytywanie kształtu oraz jego konwersję do maski. Wynik oceniano porównując stereogram z wszytanymi bitmapami. Następnie wygenerowane stereogramy zapisywano sprawdzając poprawność zapisu.

Testowanie procedur podświetlających, rysujących pomocnicze kropki, zmieniających rozdzielczość, odcięcie i kolory. Testy te odbywały się analogicznie do poprzednich. Podczas nich ponownie testowano poprzednie bloki w celu sprawdzenia integralności całego programu, co ostatecznie doprowadziło do testów całościowych na 2 załączonych bitmapach testowych.

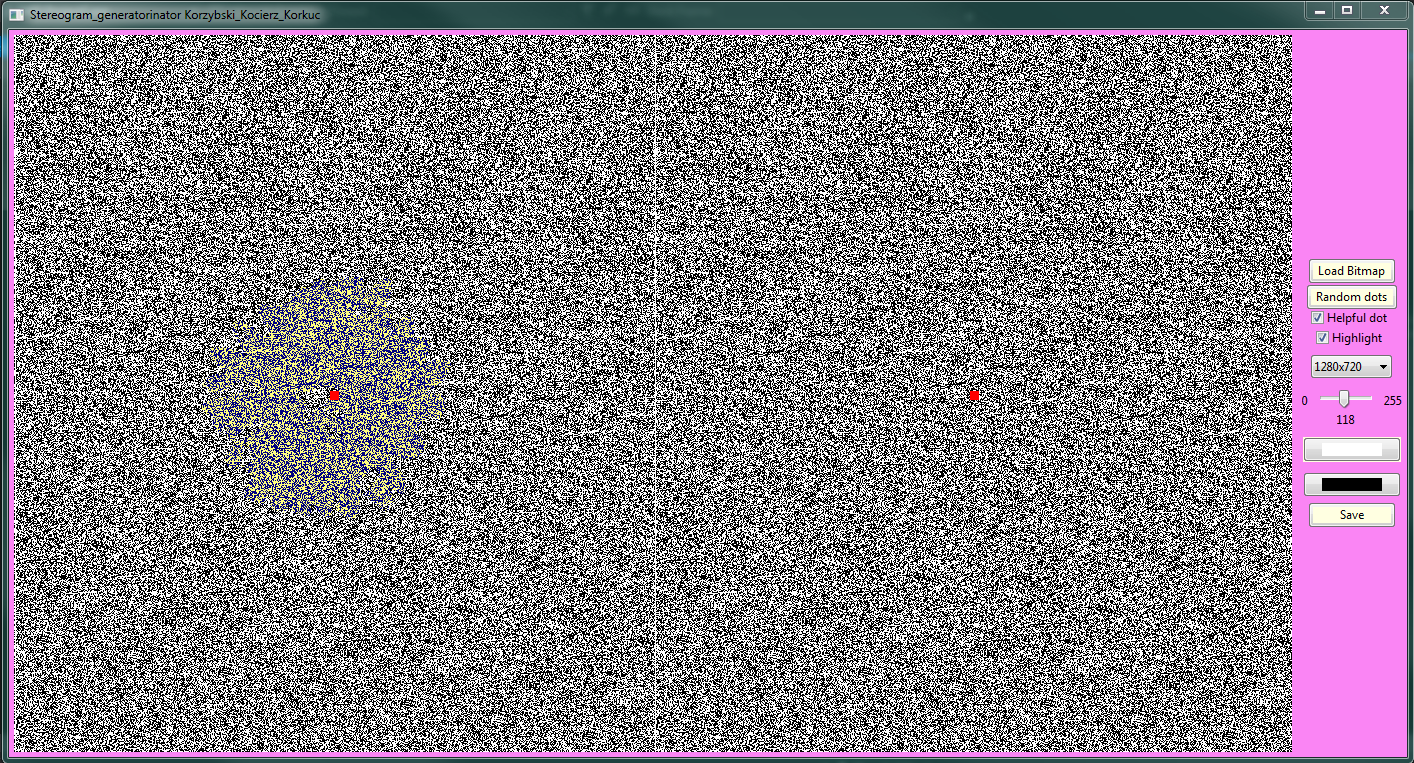
1. Testowanie połączeń między poszczególnymi blokami. Większość funkcji komunikuje się z głównym blokiem programu (funkcją rysującą) za pomocą flag. Dlatego przeprowadzono testy polegające na każdorazowym zatrzymaniu programu na początku metody draw po wykonaniu testowanego bloku w celu sprawdzenia stanu poszczególnych flag.

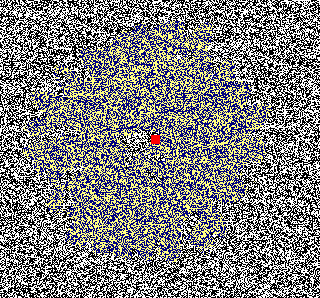
Podczas wykonywania testów naprawiano wszelkie pojawiające się błędy, co doprowadziło do bezbłędnego działania programu dla bitmap testowych. Ostatnim krokiem było przeprowadzenie testów całościowych na trzeciej bitmapie, która nie została użyta podczas wcześniejszych testów.

# Wdrożenie, raport, wnioski

Skończony program uruchomiono i załadowano kilka map głębokościowych znalezionych w Internecie.

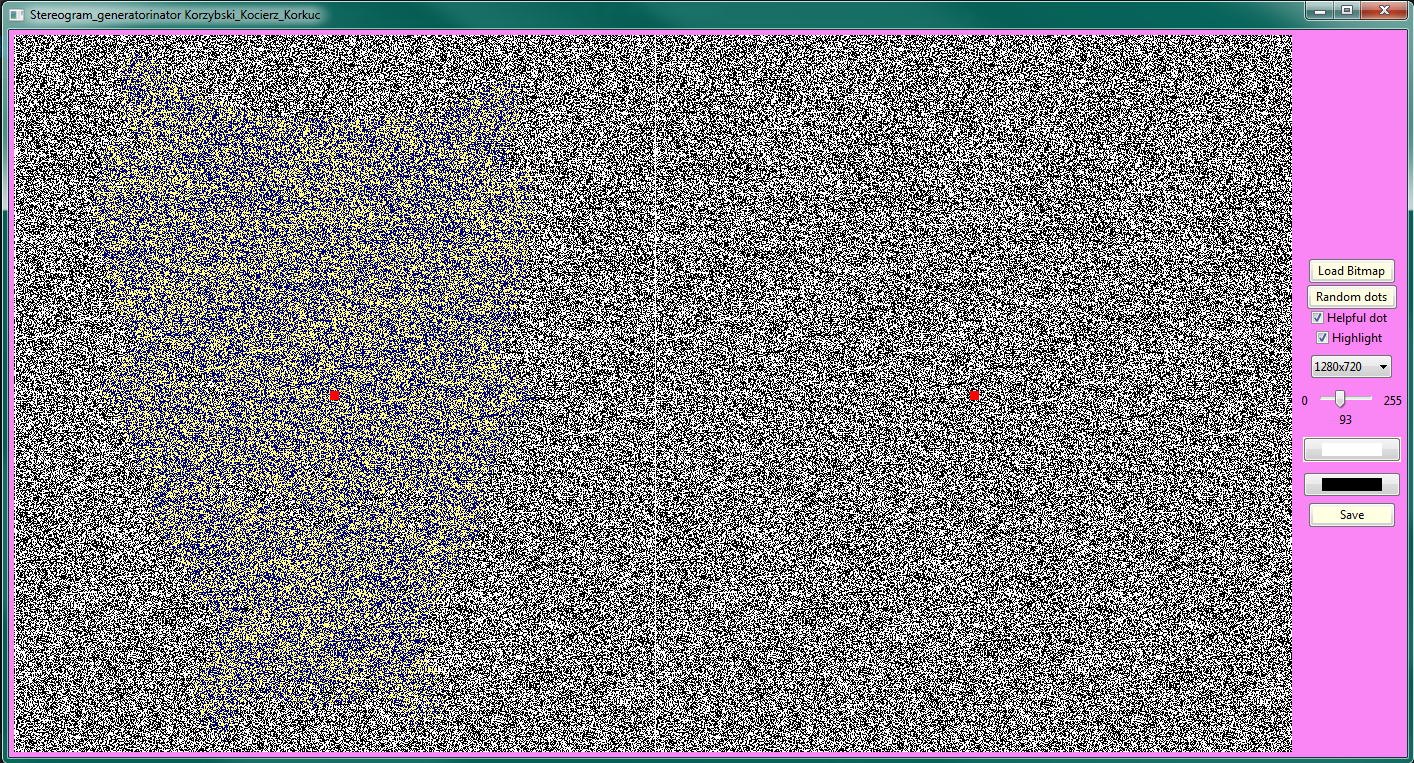
**Kwiat:**

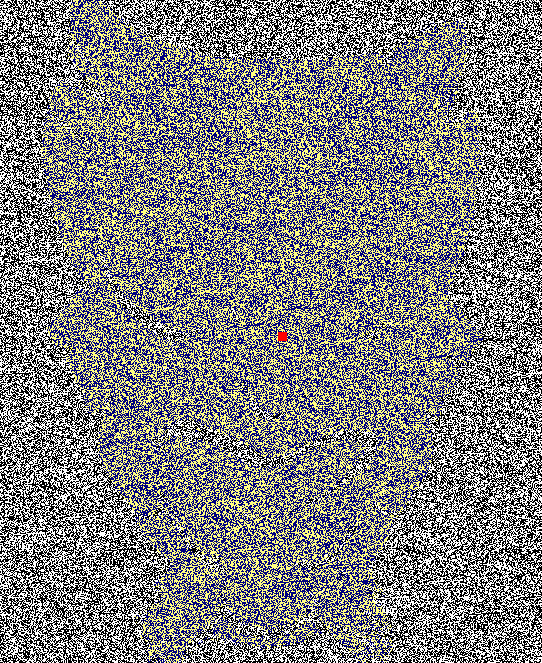


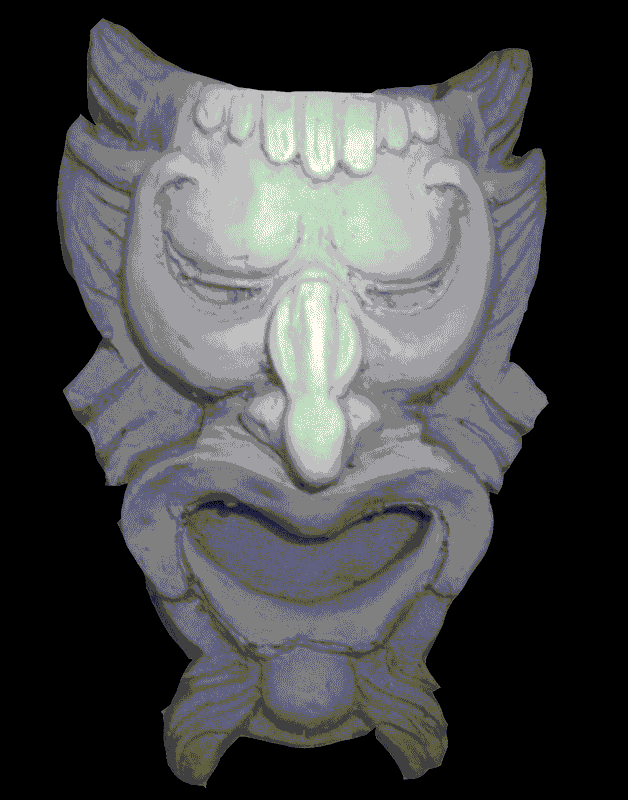




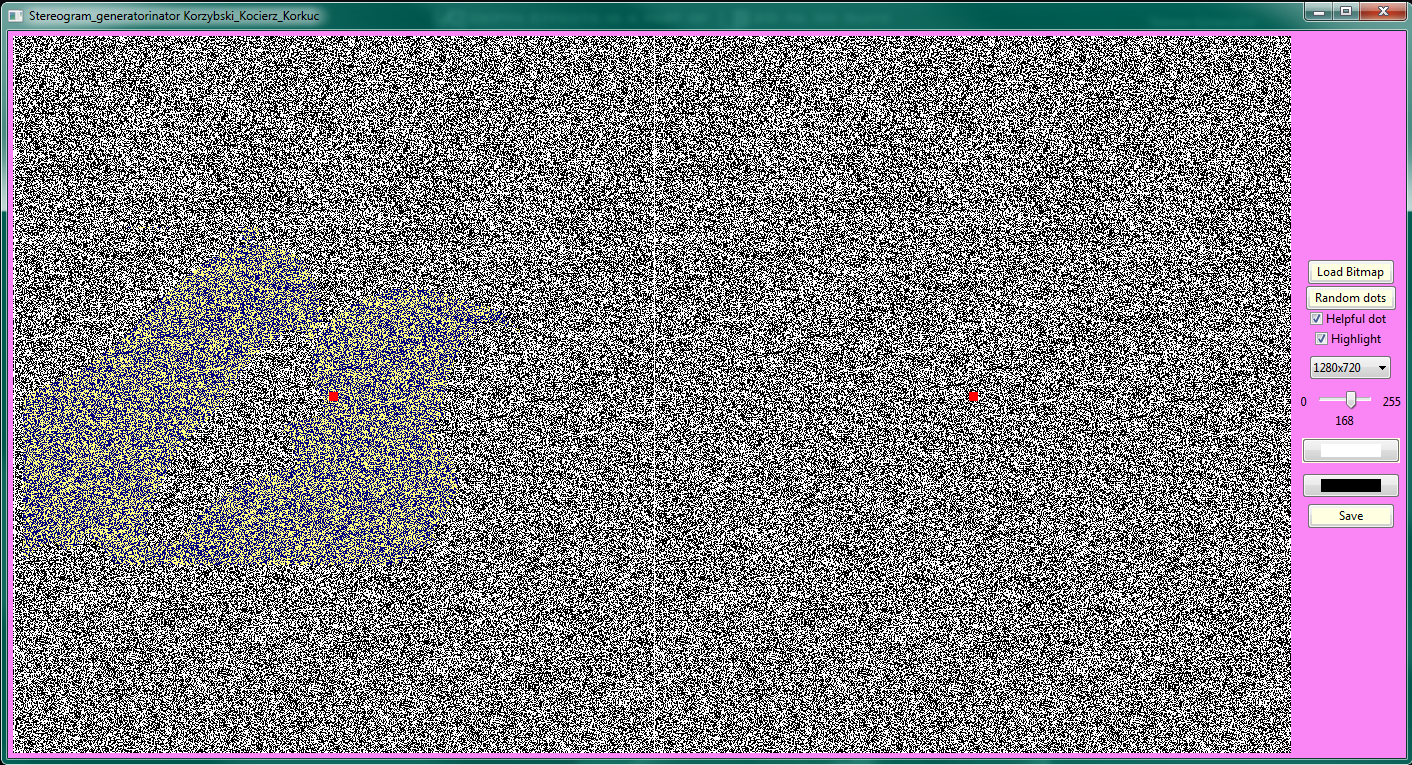
**Maska:**

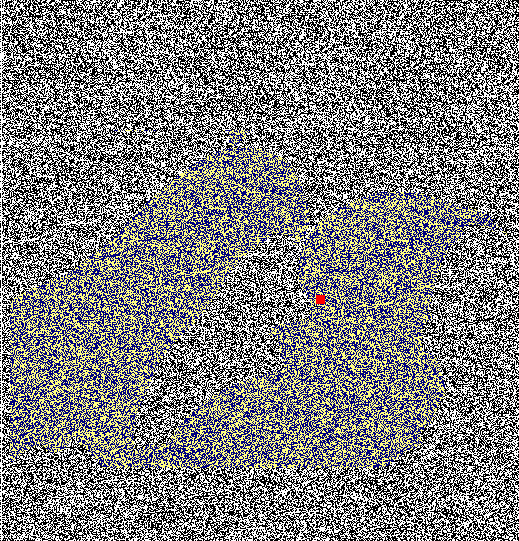


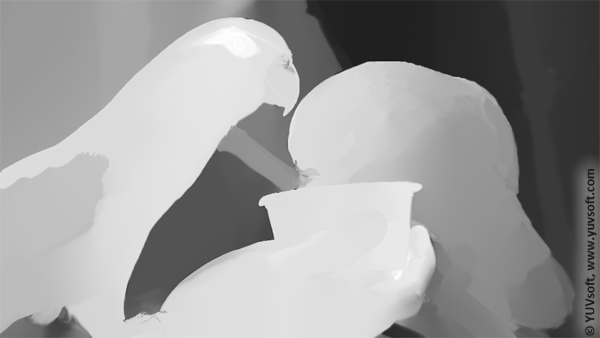


****

**Papużki:**





****

Wszystkie zaplanowane funkcje programu udało się poprawnie zaimplementować. Program działa sprawnie i nie przejawia wycieków pamięci. Możliwości rozszerzenia programu to na przykład dodanie wielokolorowego losowania kropek co mogłoby nawet zamienić się w stereogram przedstawiający jeden obiekt jako wypukły a sam obraz 2d byłby zrobiony z innych obrazów. Bardziej ambitnym projektem byłoby odtwarzanie video jako stereogram. Niestety brak video w formacie mapy głębokościowej mocno utrudnia to zadanie.