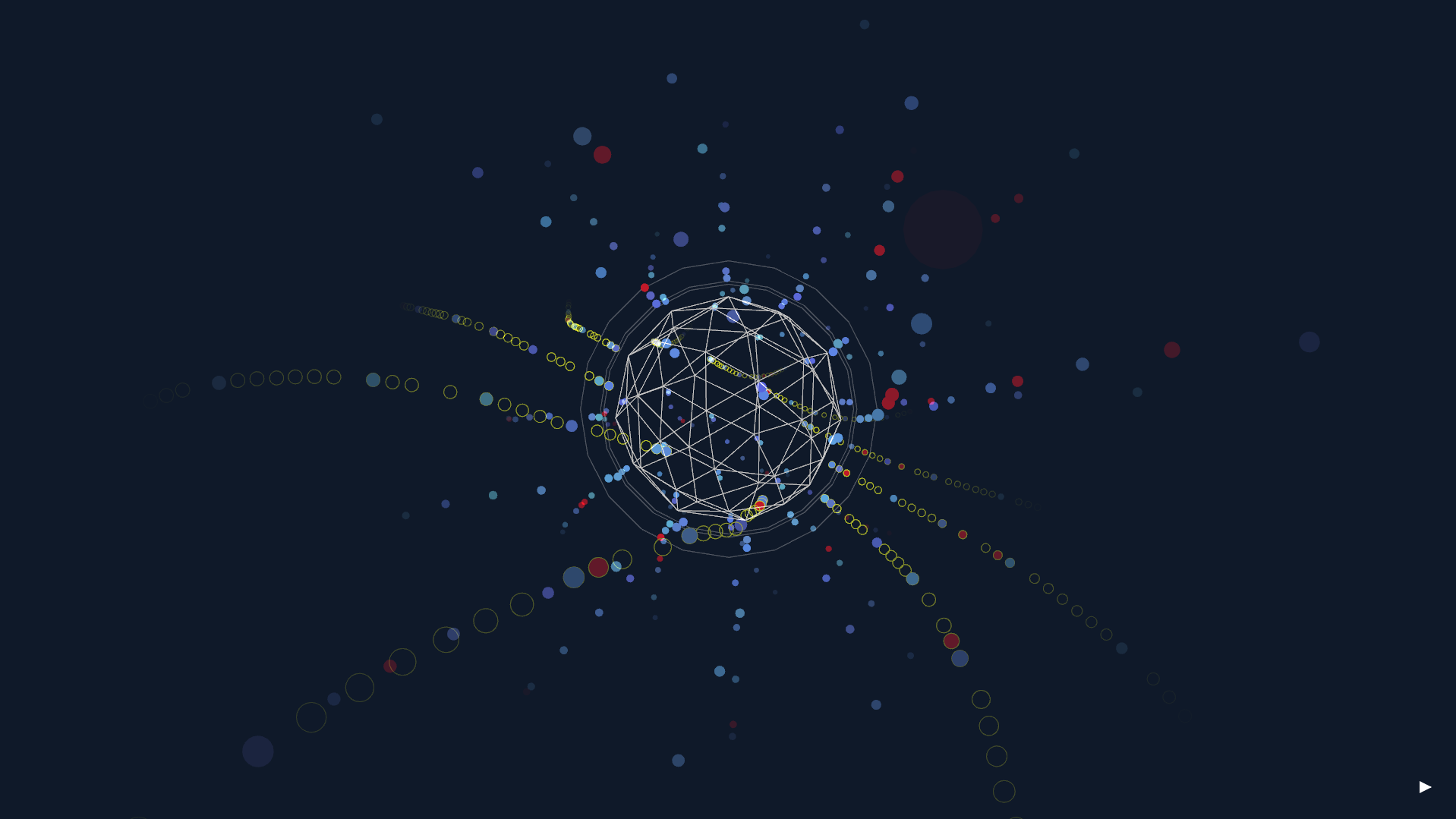
System cząsteczkowy   
z wizualizacją muzyki



**GitHub**: <https://github.com/vanbear/ParticleSystem>  
**YouTube:** <https://www.youtube.com/watch?v=K45HHaaCpVg>

# Początkowe założenia:

* stworzyć system cząsteczkowy przy użyciu OpenFrameworks
* reakcja programu na muzykę
* obracająca się kula emitująca cząsteczki w rytm muzyki

# Co udało się zrobić?

Wszystko!

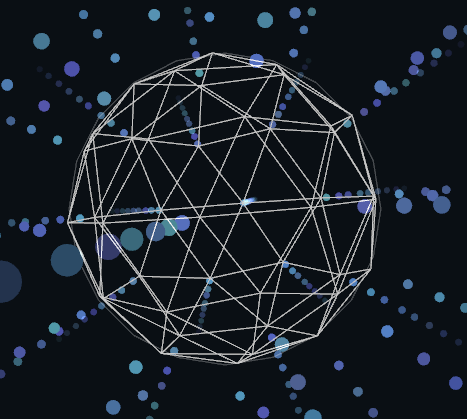
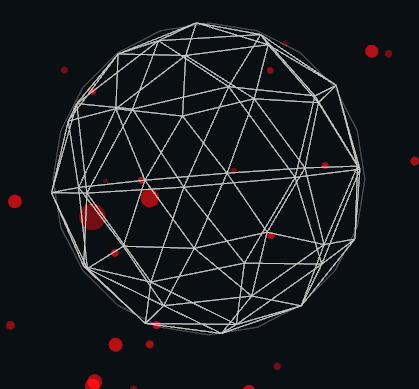
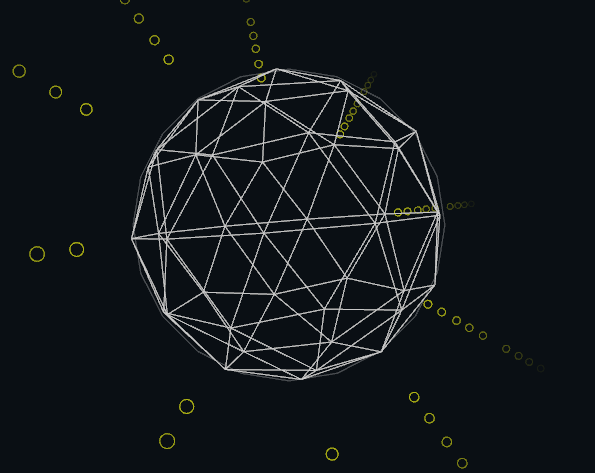
# Ważne informacje:

* Klawisz P uruchamia muzykę
* Klawisz D uruchamia tryb debugowania
* Możliwe jest załadowanie własnego pliku muzycznego po wciśnięciu klawisza L. Trzeba niestety przy tym wcisnąć Alt+Tab aby przenieść się na okno dialogowe.

# Opis projektu:

Kula na środku ekranu jest obiektem typu **ofIcoSpherePrimitive,** składa się z trójkątów, na których wierzchołkach ułożone są obiekty klasy **Emitter**, które przy spełnionych odpowiednich warunkach emitują różne rodzaje cząsteczek.  
Każdy rodzaj cząsteczki dzierżawi z klasy bazowej **Particle.** Różnią się od siebie jedynie zawartością konstruktora i funkcji draw();

**Rodzaje cząsteczek:**

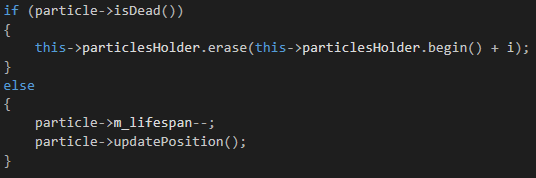
* niebiesko-zielonkawe (odcień losowy) wypełnione okręgi emitowane przez wszystkie emitery  
  
* czerwone wypełnione okręgi emitowane w N losowych miejscach  
  
* żółte niewypełnione okręgi emitowane przez emitery znajdujące się na jednej płaszczyźnie  
  

Wszystkie cząsteczki przy tworzeniu otrzymują prędkość skierowaną na zewnątrz od środka ekranu.



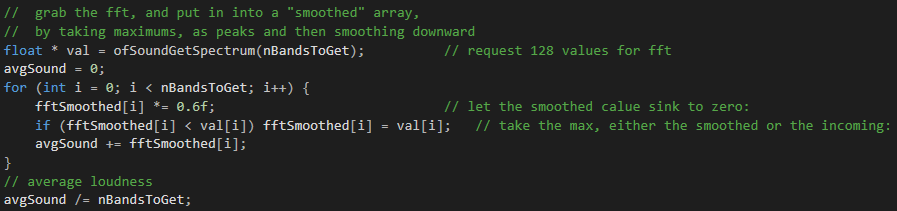
Posiadają zmienną **m\_lifespan**, która określa ich długość życia. Z każdą klatką jest zmniejszana o 1 i jest używana m.in. do określenia przeźroczystości przy rysowaniu, dzięki czemu zanikają z czasem, a bo obniżeniu do zera są niszczone.



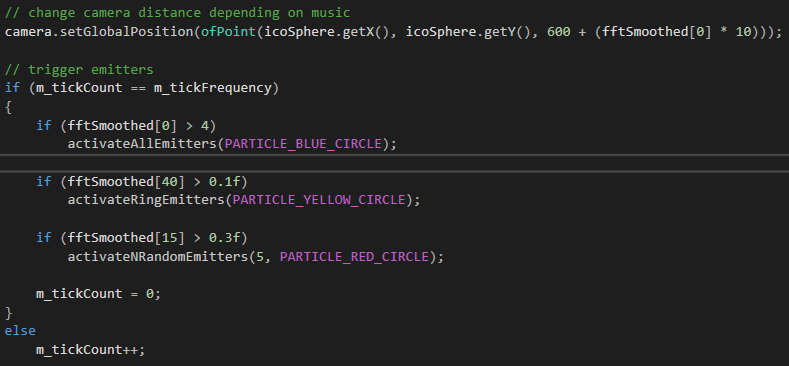


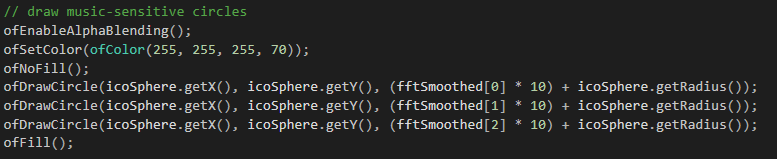
**Obsługa muzyki:**

Openframeworks posiada wbudowaną funkcję, która wykorzystując szybkie przekształcenie Fouriera zwraca tablicę o rozmiarze „**nBandsToGet**” informacją na temat sygnału dźwiękowego na różnych częstotliwościach w danym momencie. Otrzymane informacje od razu wykorzystuję do obliczenia średniej głośności dźwięku **avgSound.**



Sprawdzając określone komórki tablicy **fftSmoothed** aktywuję określone emitery oraz zmieniam położenie kamery tworząc efekt turbulencji oraz wykorzystuję do określania rozmiaru pierścieni rozszerzających się wokół sfery.





Wartość **avgSound** wykorzystuję do zmiany koloru tła oraz do wprawiania sfery w ruch





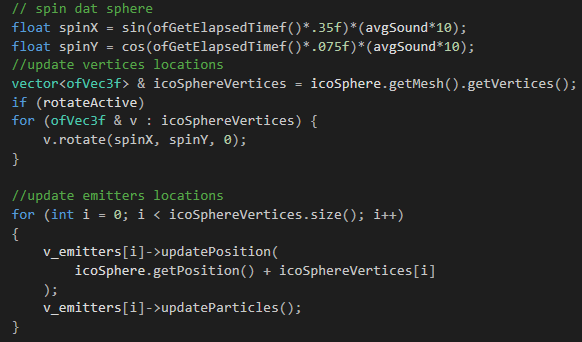
# Napotkane problemy:

* utrzymanie położenia emiterów na wierzchołkach obracającej się sfery
* odczytanie, które emitery znajdują się na jednej płaszczyźnie przecinającej sferę w pół
* analiza otrzymanych danych na temat dźwięku oraz podpięcie ich pod działanie aplikacji

# Rozwiązanie problemów:

**Utrzymanie położenia emiterów na wierzchołkach obracającej się sfery.**

Problem polegał na tym, że emitery mimo obracania sfery nie obracały się razem z nią. W każdej klatce pobierałem położenia wierzchołków sfery i przypisywałem je poszczególnym emiterom. Aktualny kod wygląda tak:

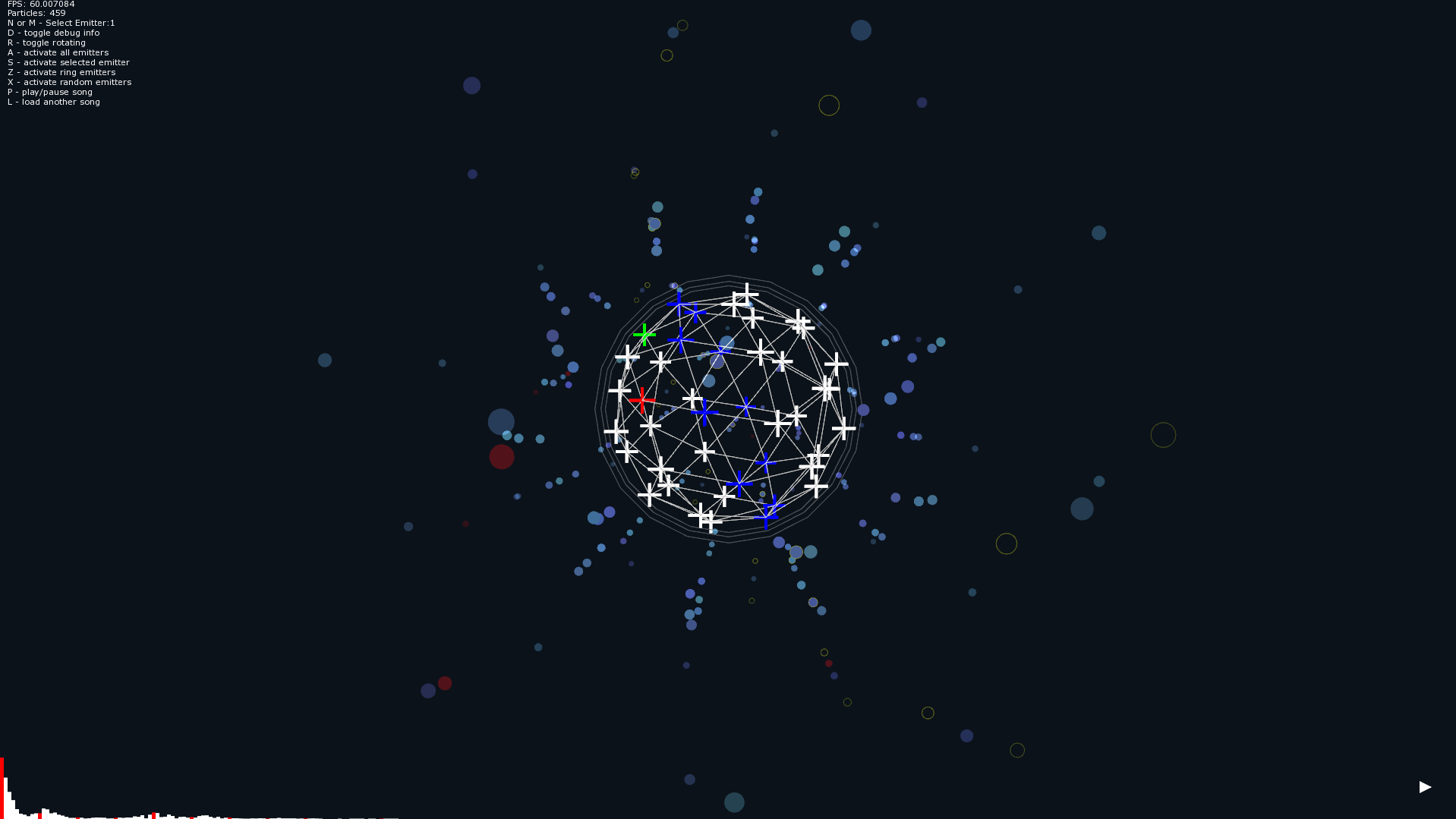


Problemem było pobieranie położeń jako wartość, nie jako referencja, przez co emitery obracały się o jeden krok i natychmiastowo dostawały położenie początkowe, które nie było w ogóle modyfikowane, przez co emitery zostawały w miejscu. Dodanie jednego „&” rozwiązało problem.

**Odczytanie, które emitery znajdują się na jednej płaszczyźnie przecinającej sferę w pół.**

Do rozwiązania tego problemu stworzyłem cały system debugowania. W lewym górnym roku są podstawowe informacje i klawiszologia. W dolnym rogu znajduje się wizualizacja tablicy fftSmoothed użyta do rozwiązania następnego problemu. Co nas teraz interesuje to emitery rysowane jako grube plusy. (screenshot poniżej)   
Te, których szukałem, znajdujące się w jednej płaszczyźnie są niebieskie. Zielony znajduje się w indeksie 0 (wszystkie emitery umieszczone są w wektorze). Czerwony to aktualnie zaznaczony. Zwiększając indeks patrzyłem który to jest emiter, jeżeli był jednym z tych, które mnie interesowały, zapisywałem jego indeks, po czym wszystkie te indeksy wrzuciłem do tablicy, do której odnoszę się np. przy emitowaniu żółtych cząsteczek. Jest ich tyle ponieważ najwidoczniej niektóre wierzchołki sfery stworzone są po kilka razy.





**Analiza otrzymanych danych na temat dźwięku oraz podpięcie ich pod działanie aplikacji**

Problem ten rozwiązałem metodą prób i błędów obserwując wykres tablicy fftSmoothed rysowany w trybie debugowania. Pozwoliło mi to na określenie w przybliżeniu przy jakich dźwiękach które wartości „reagują” najmocniej. Niestety przez „sztywną” naturę tego rozwiązania program będzie lepiej lub gorzej działał przy różnych piosenkach.