**Załącznik nr 2 do SIWZ DZP.381.21.2015.UG**

# Szczegółowy opis przedmiotu zamówienia

Przedmiotem zamówienia jest dostarczenie (opracowanie) oprogramowania umożliwiającego wizualizowanie danych w czasie rzeczywistym w przestrzeni trójwymiarowej (dalej: wizualizator). Rozwiązanie powinno opierać się o najnowsze technologie przeglądarkowe (canvas + webgl) i korzystać ze sprzętowego wsparcia wyświetlania grafiki 3d.

Rozwiązanie musi się opierać o otwarte oprogramowanie, aktualnie rozwijane lub utrzymywane przez społeczność (np. data ostatniej zmiany poniżej 1roku). Podstawowym użyciem wizualizatora jest prezentacja wielowymiarowych danych w systemie Jupyter-notebook. Wykonawca powinien dostarczyć także kompleksowe rozszerzenie do tejże platformy, które integruje z nim wizualizator danych oraz umożliwia przekazywanie do niego danych z Jupyter'a w wygodny sposób. W szczególności należy zaprojektować komunikację pomiędzy elementami interaktywnymi - https://github.com/ipython/ipywidgets/.
Niezależnie, architektura wizualizatora powinna umożliwiać jego użycie poza jupyter-notebookiem.

Typy danych możliwych do wizualizacji:

1. Obiekty statyczne - Wczytywanie geometrii w formacie danych STL (STereoLithography)
2. Etykiety tekstowe położone w dowolnym punkcie sceny
3. Wektor - Narysowanie wektora o zadanym punkcie początkowym i kierunku wraz z jego oznaczeniem
4. Pole wektorowe - Narysowanie pola wektorowego w R2 oraz R3, wraz z opcjonalnym oznaczeniem każdego z wektorów
5. Punkty - Wizualizacja punktów, wraz z opcjonalnym oznaczeniem (etykieta)
6. Linie - Wizualizacja linii złożonej z n węzłów, wraz z opcjonalnym jej oznaczeniem (etykieta)
7. Tekstura na obiekcie prostokątnym w 3D
8. Powierzchnie - Wizualizacja powierzchni opisanej jako tablica wysokości w punktach (x,y) należących do siatki regularnej
9. Niestrukturalne siatki powierzchniowe – służące do wizualizacji izopowierzchni funkcji skalarnych w 3 wymiarach
10. Woksele (volumetric element) - Wizualizacja wokseli opisanych jako tablica trójwymiarowa wartości z których każda może być mapowana na kolor, a 0 jako brak woksela.

Każdy typ obiektu możliwy jest do umieszczania wielokrotnego w obrębie jednej wizualizacji. Umieszczanie obiektu na scenie powinno być możliwe wraz z określeniem pozycji referencyjnej oraz skali i obrotu. Oprogramowanie powinno stosować metody optymalizacji w celu zapewnienia szybszej wizualizacji.

Zasilanie danych do wizualizatora ma być realizowane na dwa sposoby. Pierwszym z nich jest statyczne przekazanie odpowiednich danych. Dodatkowo wizualizator powinien udostępniać interfejs programistyczny (ang. API), który umożliwiać powinien aktualizacje danych bez konieczności ponownego inicjalizowania wizualizatora. Wraz z oprogramowaniem należy dostarczyć dokumentacje do tego interfejsu wraz z przykładami.

Wizualizator powinien umożliwiać manipulacje widokiem trójwymiarowym – zmianę powiększenia, zmiany położenia kamery. Powinien też w przypadku wizualizacji wokseli umożliwiać tryb edycji. W trybie edycji możliwe jest zaznaczanie wokseli za pomocą określonego zbioru (np. sześcian czy kula w przestrzeni wokselowej) wokseli i nadawanie im określonej wartości. Po edycji danych powinno móc dać się wyeksportować nowy zbiór wokseli do postaci tablicy trójwymiarowej, w formacie “npz”, z możliwością zapisu na lokalny system plików i alternatywnie przekazania tych danych do uruchomionego procesu pythonowego z wykorzystaniem komunikacji sieciowej. To ostatnie rozwiązanie jest elementem integracji trybu edycji z notatnikiem jupyter.

Rozwiązanie powinno być otwarte na łatwą możliwość rozszerzenia go w przyszłości o nowe typy obiektów, które potrafi wizualizować. Niezbędnym elementem jest stworzenie elastycznego i udokumentowanego API komunikacji – wizualizator - proces dostarczający dane, oraz rozdzielenie procesów samej wizualizacji przetwarzania i komunikacji sieciowej. Wymagane jest zastosowanie otwartej biblioteki threejs jako warstwy wizualizacyjnej. W projektowanej architekturze wizualizatora, należy jednak stworzyć możliwość przyszłej wymiany tej biblioteki na inną jeżeli zajdzie taka potrzeba - dlatego wymagana jest separacja zadań. Należy dostarczyć dokumentację opisującą taki hipotetyczny process wymiany backendu grafiki 3d - wskazującą na miejsca w kodzie, których należy zaimplementować odwołania do innej biblioteki.

Wymagane jest by cały proces tworzenia był na bieżąco publikowany na github.com i by kamienie milowe tego projektu zostały sukcesywnie ogłaszane na grupach ipython-dev i jupyter (co najmniej trzy posty). Do wykonawcy należy też zadanie, konsultacji ew. planów zmian API systemu jupyter notebook z twórcami, poprzez w/w listę mailingową, tak by projekt był kompatybilny z aktualnymi wersjami developerskimi ipython oraz jupyter notebook na miesiąc przed zakończeniem prac.

Zakres zamówienia obejmuje również:

- Wykonanie instalacji w środowisku UŚ
- Pozytywne przeprowadzenie testów funkcjonalnych w środowisku UŚ
- Przekazanie dla UŚ kompletnej dokumentacji. Dokumentacja powinna uwzględniać opis poszczególnych modułów, ich strukturę, interfejs wymiany informacji oraz wymagania sprzętowe i programowe, które powinno spełniać środowisko. Dokumentacja techniczna API jak i wszystkie komentarze wewnątrz kodu mają być w języku angielskim, tak by możliwy był dalszy rozwój kodu przez społeczność międzynarodową. Dokumentację należy przekazać w formie elektronicznej w wersji edytowalnej oraz nieedytowalnej (.pdf). Dodatkowo dokumentacja powinna być opublikowana na serwisie github.

Zamawiający nie określa środowiska i miejsca wykonywania pracy, wymaga natomiast, aby oprogramowanie będące przedmiotem zamówienia zostało zainstalowane, uruchomione i testowane w środowisku Zamawiającego.

Ostateczna postać oprogramowania, kod źródłowy i dokumentacja mają zostać:

- przekazane Zamawiającemu w postaci repozytorium kodu źródłowego wraz z dokumentacja instalacji i użycia w postaci elektronicznej

- zainstalowane na systemie jupyter notebook dostarczonym przez Zamawiającego.
- opublikowane na serwisie github, jako repozytorium kodu.
- proces instalacji i uruchomienia oprogramowania musi być zademonstrowany z użyciem jedynie repozytorium github i komend dostępnych w systemie Linux.
- kod źródłowy musi być czytelny, dokumentacja, komentarze i nazwy zmiennych i obiektów w kodzie muszą być w języku angielskim,
- kod musi być przystosowany do dalszego rozwoju przez społeczność. Kod należy udostępnić na licencji Gnu Public Licence version 2, lub kompatybilnej.