1. Course Description

a. Title of a Course

Analysis and Visualization of Networks

b. *Pre-requisites*

(B.1) Programming basics.

(B.2) Algorithms and data structures basics.

(B.4) Applied graph theory.

Real programming experience is eligible.

c. *Course Type*

Elective

d. *Abstract*

This course introduces methods and algorithms for analysing and visualizing graphs and networks. The course includes a review of modern network analysis and visualization techniques with their applications in various domains. We will concern on three main topics: network analysis methods based on applied graph theory, graph drawing algorithms, applications of network analysis and visualization to real problems.

2. Learning Objectives

By the end of this course students:

a) will *know* the classification of main network analysis tasks, basic methods and algorithms, most popular software tools;
b) will be able to *define* a graph-theoretic description of network analysis task and corresponding network visualization requirements;
c) will be able to *select* reasonably an appropriate project solutions and tools for network analysis workflow;
d) will be able to *develop* a new variants of graph drawing algorithms.

3. Learning Outcomes

Upon successful completion of this course, students will demonstrate:

a) an ability to design and solve graph-theoretical mathematical models;
b) an ability to select and justify appropriate graph drawing method and algorithm;
c) an ability to use development techniques, skills and tools necessary to network analysis;
d) an ability to use development techniques, skills and tools necessary to network visualization;
e) an ability to communicate effectively;
f) an understanding of professional and ethical responsibility.

4. Course plan

Lectures: 22 academic hours.

Seminars: 22 academic hours.

Self-training: 64 academic hours.

1. Introduction
   1.1. The classification of graph analysis tasks.

1.2. Main approaches to graph algorithms.
1.3. Graph data file formats.
1.4. Graph databases.
1.5. The pool of main network analysis tools.

2. Graphs, topology and geometry.
   2.1. Adjacency and neighbourhood.
   2.2. Hierarchies, trees and taxonomies.
   2.3. Cliques and dense fragments.
   2.4. Centrality.
   2.5. Planarity.
3. Visualization of small graphs: drawing and layout.
   3.1. The classification of goals and constraints.
   3.2. Symmetry-based approaches.
   3.3. Hierarchical approaches.
   3.4. Iterative approaches.
   3.5. Force-directed drawing.
   3.6. Orthogonal drawing.
   3.7. Radial and circular drawing.
   3.8. Treemaps.
   3.9. Geographic layout and maps.
4. Visualization of large graphs
   4.1. Scalability.
   4.2. Graph fragments and filters.
   4.3. Approximate drawing.
   4.4. Random walks and other randomization techniques.
5. Interactive visualization of graphs.
   5.1. Zoom, scale, pan, rotate.
   5.2. Dynamic visualization.
   5.3. Best practices in user interaction.
6. Visualization of graphs and networks in real world applications.
   6.1. Social networks analysis.
   6.2. Logistics and supply chains.
   6.3. Cheminformatics.
   6.4. Bioinformatics
7. Modern trends in graph databases and network analysis software.

5. Reading list

a. Required

1. *Handbook of Graph Drawing and Visualization*. Edited by Tamassia R. CRC Press, 2013. 862 p.
2. Brath R., Jonker D. *Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data*. Wiley, 2015. 513 p.
3. Russell M.A. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. 2nd ed. O'Reilly, 2013. 448 p.
4. Awesome-network-analysis – A curated list of awesome network analysis resources. Web. (http://github.com/briatte/awesome-network-analysis)
5. The GraphML File Format. Web. (http://graphml.graphdrawing.org)

b. Optional

1. Steen M. *Graph Theory and Complex Networks: An Introduction*. 2010. 300 p.
2. Ahuja R.K., Magnanti T.L., Orlin J.B. *Network Flows: Theory, Algorithms, and Applications*. Pearson, 1993. 864 p.
3. Jackson M.O. *Social and Economic Networks*. Princeton University Press, 2010. 520 p.
4. McGuffin M.J. *Simple Algorithms for Network Visualization: A Tutorial*. Tsinghua Science and Technology, 17(4), 2012, pp. 383-398.
5. *Graph Drawing*, 5st – 25th International Symposium, 1997-2007 (See http://dblp.uni-trier.de/db/conf/gd/ and http://graphdrawing.org/symposia.html).
6. The House of Graphs. Web. (http://hog.grinvin.org)

6. Grading system

Students' final grades are based on the following activities: *home assignments* at seminars, *in-class assignment*, *individual project*.

Ongoing assessment is delivered as *home assignment* (HA), *in-class assignments* (ICA) and *individual project* (IP). Grades, which are gained by students while ongoing assessment, are parts of the *cumulative grade* (CG). Final grade (FG) is calculated as follows: FG = CG.

$$FG = 0{,}15 \cdot HA_1 + 0{,}15 \cdot HA_2 + 0{,}15 \cdot HA_3 + 0{,}1 \cdot ICA + 0{,}4 \cdot IP$$

Only rounded grades take place in calculations. The arithmetic rounding-off rules are used. Example: $4.5 \rightarrow 5$; $4.49 \rightarrow 4$.

Attendance is graded according to 10-point scale applied in NRU HSE.

10-point scale	Russian grading framework	ECTS grading scheme	
10	*Excellent*	**A+**	*Excellent*
9	*Excellent*	**A**	*Very good*
8	*Excellent*	**A–**	*Very good*
7	*Good*	**B+**	*Good*
6	*Good*	**B–**	*Good*
5	*Satisfactory*	**C+**	*Satisfactory*
4	*Satisfactory*	**C–**	*Satisfactory*
3	*Fail*	**F**	*Fail*
2	*Fail*	**F**	*Fail*
1	*Fail*	**F**	*Fail*

7. Guidelines for Knowledge Assessment

Ongoing assessment

***Home assignment* (*HA*)** is an individual programming assignment which is targeted to force students to familiarize with Unreal Engine 4 and help them to shape their skills in the UE4 editor. A student prepares home assignment for 2-3 weeks (this depends on the difficulty of an assignment) and presents the results of his/her work in-class.

Deadline is soft, and a student may introduce his/her home assignment for two weeks after the soft deadline. The home assignment evaluates with 50% penalty. There is a *hard deadline* after these two weeks. There is no possibility to gain a mark after the hard deadline.

Not working program code evaluates as a fail (2).

***Individual project* (*IP*)** is implemented as a course project, which is prepared in the form of applied project with a final report and in-class presentation of results. The report contains a problem statement, a software requirements specification (SRS), project solutions, implementation details, correctness analysis with test cases and conclusion.

Deadlines are hard.

A teacher evaluates reports and provides each group with a *formative feedback*. Every member of a team gains the same grade. Student who missed a public presentation does not gain a mark.

***In-class assignment* (*ICA*)** is a group assignment. The assignment is implemented in form of bibliography and project analysis. ICA should support IP by best cases and practices.

IP report structure and requirements:

a. *Project name*. Make up a name for your network visualization and analysis task.
b. *Problem statement*. In a few sentences describe a domain, statement and constraints.
c. *Software Requirement Specification*. At least define some functional and nonfunctional requirements.
d. *Project solutions*. List the solutions, which follows from your requirements.
e. *Development tools*. Describe the developer's tools set you use to implement algorithms.
f. *Algorithms and complexity*. Describe the implementation of the algorithms and give the theoretical and empirical analysis of their computational complexity.
g. *Correctness analysis*. Give a test cases and justify correctness of your solutions.
h. *Conclusion*. Describe maturity of your solutions and potential future plans.

Assessment schedule

Week	Assessment type
***Module* 3**	
Week 3	HA_1 deadline
Week 5	HA_2 deadline
Week 6-7	In-class assignment (ICA)
Week 9	HA_3 deadline and IP preliminary report deadline
Week 10-11	Individual project deadline and presentation (IP)

8. Methods of Instruction

The course combines direct and indirect teaching techniques and contains: lectures, collaborative learning, problem based learning, blended learning, reports, feedback and formative assessment.

Direct teaching is carried out by a lecture method. The preferred informal lectures (lectures with discussion), where students play an active role. The main purpose of lectures is to introduce course's topics, to overview basic information and to discuss and form the directions of further course activities.

Problem based learning is implemented through *home assignments* and *individual project.*

Home assignment means the individual work of a student on specific methods of graph drawing in form of programming assignments.

Individual project means the individual work of a student on applications of graph drawing in form of applied project.

In-class activity (*seminars*) consists of short programming assignments and discussions. Students works in pairs (*think share pare technique*) or in small randomly generated groups.

Blended learning is implemented through the *collaboration* and *formative assessment* in online/offline students' activities using collaborative spaces powered by the Microsoft Collaborative Platform (Microsoft Office 365 and OneNote Class Notebook).

9. Special Equipment and Software Support

Equipment for in-class and home activity:

1. Classroom multimedia projector;

Software recommended for in-class and home activity:

1. Microsoft Office 365.
2. Web-browser with HTML5 and JS support.
3. Jupiter Notebook with Python kernel.
4. Microsoft Visual Studio 2015 or upper.
5. yWorks yEd Graph Editor (https://www.yworks.com/products/yed).
6. Gephi (https://gephi.org).
7. UCINET Software (http://sites.google.com/site/ucinetsoftware/home)