Basic Information

Project Name	Domov
Abbreviation	DOMOV
Supervisor	Martin Dlask <martin.dlask@bisimulations.com>
Consultant	Jakub Gemrot <jakub.gemrot@gmail.com>
Team Members	Jakub Svoboda (MFF UK), Jan Kočur (MFF UK), Ekaterina Bessonova (FAMU), Marius-Alexandru Anagnoste (MFF UK), Yulia Malkova (MFF UK)
Annotation	Domov is a low poly 3D survival game where the player tries to survive in a living ecosystem. The player must satisfy his needs, explore the environment and find ways of making life easier. The player must be cautious though, because if he destroys the ecosystem his own doom follows.

Motivation

A genre of games that gained popularity in the last few years is the survival games genre. The most successful games of this genre include: Minecraft, Don't Starve, Rust, The Forest. In these games, the player is usually tasked with fulfilling his needs like hunger, sleep, stress, etc. The games usually focus on surviving, exploring and crafting better tools.

What survival games seldom concern themselves with is the player's influence on the world he is living in. The idea, that the actions of a single person can leave the environment in balance or bring about its possible downfall, is not represented in any survival game as the core concept that we know of.

This kind of topic is very relevant for the present and near future and we believe that games have the potential to convey such an idea with greater efficiency than other traditional media.

During the course Computer games development, we developed a prototype of the game Domov. In this prototype, we explored the idea of player's consequences on the world and learned that it is possible to make this idea the core concept of the game. That's why we decided to continue the development of Domov into a more complete and finished game.

Project Description

The goal of the project is to:

- Do game design to specify all the fundamental features and use-cases
- Design the software architecture of the game
- Implement the features according to the architecture design and/or improve on it
- Playtest the game's core features and iterate over their design
- Design the visual style and create art assets for the game features, which are finished

In the following sections we will describe:

- Sec. 1: Overview of the game Domov. What are the intended features of Domov that we would like to see in the finished game. These will serve as the requirements for the project.
- Sec. 2: What was accomplished in the Domov prototype.
- Sec. 3:What we are going to do within this software project.
- Sec. 4: The project development cycle outline.

1. Game overview

Domov is a low-poly 3D survival game. It takes place on a flying island in space that is home to a living ecosystem. The game primarily focuses on the player exploring a living environment, player interactions with this environment and the consequences that result from these interactions.

Player

The player plays as a girl who wakes up on the island. The girl does not know who she is or where she is. The player will be able to find this out by progressing through the story, if he survives long enough.

The player survives by attending to his needs. The needs are: health, hunger, thirst, stress and warmth. Every need is different in what makes the its level increase and decrease, what the need influences and what is its feedback to the player. The feedback is done through animations, voice lines, screen effects and UI indicators.
There are two need-like mechanics in the game, that will serve as the motivator for the player to interact with most of the environment entity types:

- Illnesses: The player will contract illnesses from all kind of negative states, e.g. being wounded could mean the possibility of infection, being too cold could mean getting a fever, etc. To cure illnesses and injuries the player will have to gather resources from the environment and in some cases travel to different biomes, forcing him to interact with a greater part of the environment.
- Nutritional requirements: This is basically a sub-need of hunger. The idea is to give the player certain positive boost effects, if he is able to have a diverse diet. In the opposite case, if the player neglects his nutrition severely he will be inflicted a negative effect. This means that to get the positive effects, the player will be encouraged to hunt and gather animals and plants from all kinds of biomes, thus encouraging him to interfere with the environment and to explore it.

To make survival easier the player can craft weapons, clothes, tools, build simple structures, cook meals and prepare remedies and other medicine. The player will also be able to upgrade existing items and structures. There are several things that the player will need for crafting:

- Resources: The player will need to explore and interact with the environment in order to get these.
- Tools: Some items will require tools some will not. Tools are either small tools in the inventory (like a knife) or some large ones (like a forge or a workbench).
- Recipes: To be allowed to craft an item the player needs a recipe. The recipe also tells the player what resources and tools he will require. Recipes will be unlocked through story progression and exploration.

The player gathers resources by hunting, foraging and mining. Hunting can be done using ranged weapons and traps. Melee weapons are mostly for protection against predators or dealing with trapped animals.

The story will be told through exploration, environmental storytelling and finding logs and pieces of information throughout the island. There could also be some story-player interactions depending on the story.

There is a lot of information and feedback the player has to get and manage. Some of this feedback will be done through in-game animations, sounds, effects or even things like the number and state of plants or number of birds for the state of the ecosystem. The rest will have to be done through the UI. The more traditional elements of the UI will be things like menus and option screens. For the in-game UI we want to be more minimalistic and less intrusive on the gameplay. The girl will have a smart gadget on her wrist that she is able to look at, which will serve as the in-game UI and display her statistics, inventory and crafting screens.

Due to the game’s complexity, the game will feature a small tutorial section. In this section the player will learn the basic mechanics and will be given other useful gameplay hints.

Ecosystem

The ecosystem consists of a predator species (carnivores), a consumer species (herbivores) and a producer species (plants). The predators hunt the consumers and the consumers eat the producers. Without the influence of the player the ecosystem is balanced. In other words, there is never too many or too few of predators, consumers or producers, which would cause a collapse of the ecosystem.

There will be in total 5 biomes: fields, woodlands, wetlands, tundra and mountains. Each will have its own ecosystem with different fauna, flora and other resources.

The ecosystem will be sensitive to player’s actions. The player can influence the ecosystem in the following ways:

- Hunting predators or consumers
- Collecting resources, like mining or collecting herbs
- Player’s presence scares off animals, which can have a negative impact
- In similar way the player’s buildings will take up a radius around them, where animals won’t feel safe

The ecosystem will provide feedback about its state. There will be special plants that will respond to the balance of the ecosystem. Player will be able to tell the state of the ecosystem by looking at how much of these plants there are and what is their state (e.g. optimal, neutral, rotten). Birds are not part of any of the ecosystem core chain, but they will like the plants provide feedback about the ecosystem state. Of course the player will see the impact in the reduced numbers of the ecosystem entities as well. Another way of providing feedback could be through the usage of an indicator in the smart gadget the player character is going to have.

There will probably be more ways of providing the feedback in the future as this is one of the most important aspects of the game: the player must be able to tell that he is influencing the ecosystem and he must he must tell how he is influencing it.

The ecosystem is not going to be fully simulated as it was during the prototype. This is due to the lack of control and because the resulting balancing needed is very unwieldy. The ecosystem will still seem to be lively, but it will be controlled on several levels. This is to be able to do easier balancing, where if the player negatively influences the ecosystem, we can just reduce the number of ecosystem entities programmatically and vice versa. This will also allows us to do more complicated stuff, where we will be able to tell packs/herds to move to a certain destination or do a certain thing. Another benefit is optimization, if the player is far away from a certain location, we do not have to actually simulate all the entities in that region.

The behaviours the agents will display:
- Forming packs and herds
- Walking around
- Interaction inside the packs/herds
- Looking for food
- Hunting
- Eating
- Sleeping
- Reproducing
- Migrating from one food source to the other
- If the ecosystem inside a biome collapses, the remaining animals will spread to other biomes. Because the animals will be out of place in the other biomes, they will be able to destabilize the ecosystem inside that biome as well. This is done so that the player cannot selectively destroy biomes and get away with it.

Environment

As mentioned in the ecosystem section, the island will consist of five distinct biomes. Each biome will have its own environmental features. That includes terrain characteristics and weather effects. For example the mountains will have lots of uneven terrain and fewer passes and be much colder than the fields, which are going to have much more even terrain and neutral temperature.

The island should be large. In other words: each biome should take a few minutes to pass through when running from one end of the biome to another.

Crafting a large environment by hand is very tedious and time consuming. Also it is not flexible to changes in design. That is why we decided to implement procedural generation system into the game.

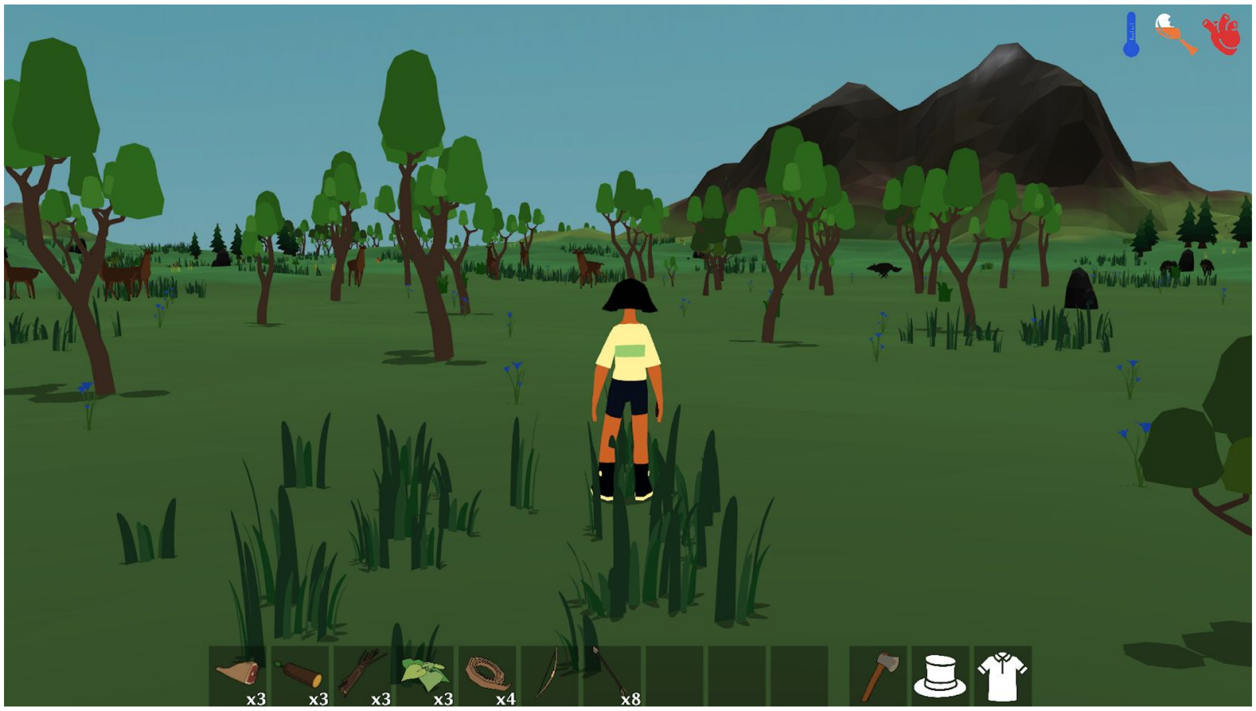
Having the island be procedurally generated also helps with replayability of the game, since large portions of the gameplay will be consisting of the player exploring his surroundings in search of food, resources for crafting and story-related objects and places. With procedurally generation, the player will be able to do this in a new environment on each playthrough

2. Domov prototype

The prototype was developed in 12 weeks. That includes the game design, art assets creation, programming, testing and balancing. Because of this lots of the implemented features are either very buggy or in poor state.

The prototype features the basics of some of the game features:

- Agent simulation. The agents are able to look for food, escape, hunt, reproduce and migrate. The agents are completely simulated, i.e. the game does not have any hidden rules that can change their behaviour. This makes them extremely hard to balance and control.
- Procedural generation is able to generate a small island with one biome and place the objects in relatively appropriate places. It is unable to generate water bodies, large island, multiple biomes and in general have more variety to it.
- The player mechanics include two needs: hunger and warmth, that are just slowly depleted over time. The player can cut wood, collect items on the ground, craft few basic items and fight animals.
- The user interface (UI) is comprised of the main menu, settings, few in-game meters, the inventory and crafting window. The UI implementation lacks any unifying architecture, making it harder to maintain, change and extend.



3. The Project

In this section we will look at what will have to be redone (from the prototype) and done in the course of the project. We will go one by one by the sections as described in the overview: Player, Ecosystem, Environment. After that there will be a general section.

Player

What will have to be mostly redone:

- Controls
- Combat
- Interactions

What will be new:

- Better animation system, e.g. the character should not glide on the environment, animation should be more responsive, etc.
- Building construction system
- Extending the crafting hierarchy
- Adding the rest of the needs: thirst, stress
- Implementing illnesses and nutritional requirements
- Implementing traps
- The smart gadget UI system: statistics, inventory, crafting and story-progression or exploration screen

- Cooking system
- Tutorial
- Story-related features

Ecosystem

What will have to be mostly redone:

- Agent behaviours: Looking for food, Reproduction, Migration

What will be new:

- Global management system for the ecosystem and agents
- Feedback system and responses to player's actions
- Agent behaviours:
    - Interactions inside herds (e.g. children-parents interactions, deer males fighting, cubs playing etc.)
    - Migration to other biomes when an ecosystem collapse occurs
    - For predators: attacking the player
- Better animation system for the agents (similar to the player's new animation system)

Environment

What will have to be mostly redone:

- The architecture of the procedural generation system, as it is not build for multiple biomes and large terrains
- Generating the shape of the island
- Keeping track of ecosystem entities
- Environment map representation
- Place trees, plants, rocks and other ambient objects around the island in a natural way
- Place animal spawning locations

What will be new:

- Procedural generation system:
    - Generate different biomes
    - Place atmospheric objects like mists or clouds
- Performance-related features that will have to be implemented:
    - Split the resulting terrain into chunks
    - Display just the chunks that are close to player
    - Use level of detail for far displayed chunks
- Spawning story related items and locations
- Spawning atmosphere related objects like clouds, mists, asteroids and other phenomena
- Environmental events, e.g. storms, fires, wind, etc.

General

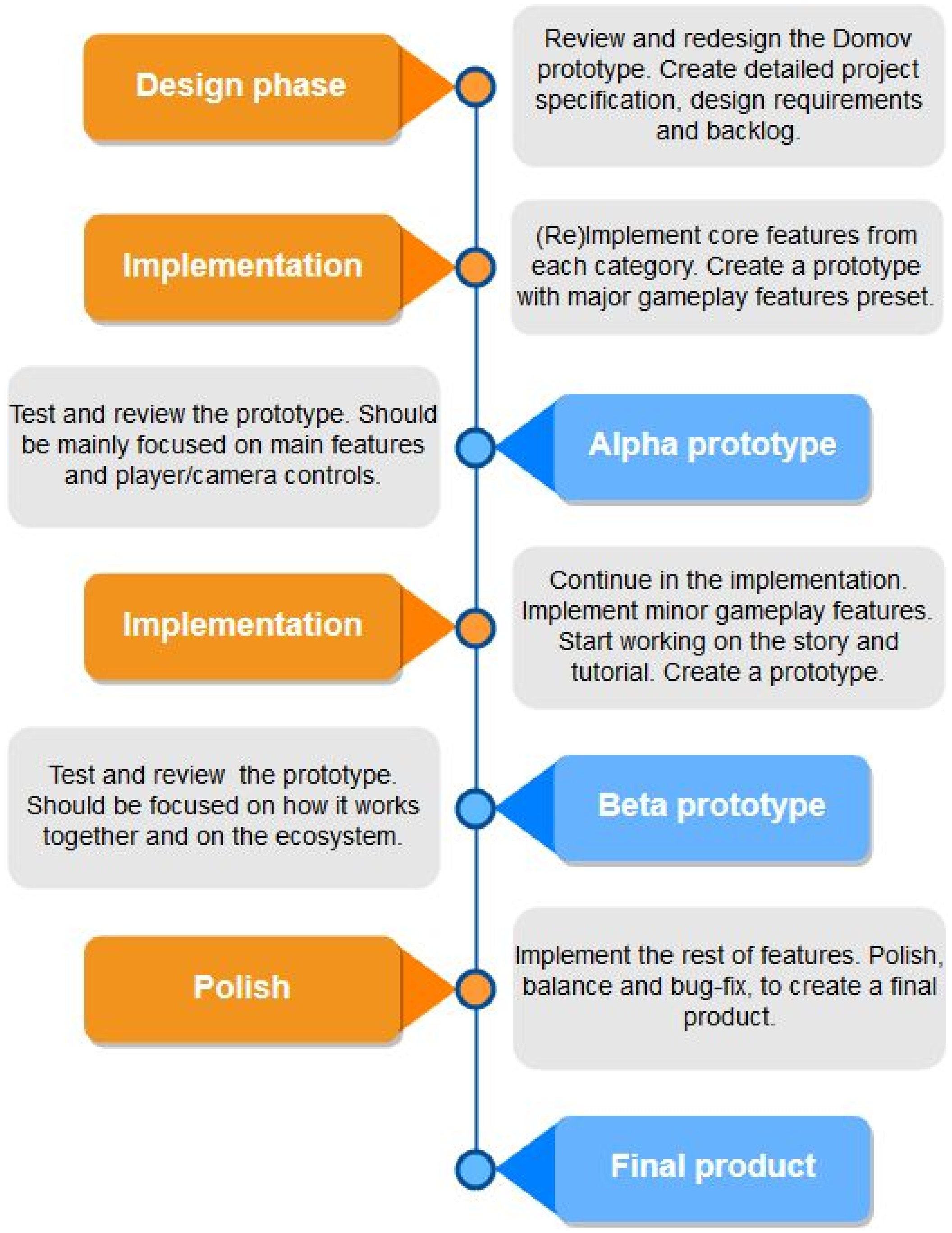
The game is not intended to be played in one session, so we will need a saving system. That means that all the relevant features that generate some data have to be designed in a way to allow for easy serialization and deserialization.

A significant portion of the time during the project will be given to play-testing,balancing the game and re-iterating on the game design. Play-testing will ensure the validity (fun) of our game design and balancing will enable us to provide a more polished gameplay experience.

4. Project outline

In the picture below we break down the project into phases.

During the development we will create two prototypes. Their goal will be to identify main problems in the project and design early on. We think that this will help the product in the long run and should give us some feedback, which is important in game development.


Design phase
Review and redesign the Domov prototype. Create detailed project specification, design requirements and backlog.
Implementation
(Re)Implement core features from each category. Create a prototype with major gameplay features preset.
Test and review the prototype. Should be mainly focused on main features and player/camera controls.
Alpha prototype
Implementation
Continue in the implementation. Implement minor gameplay features. Start working on the story and tutorial. Create a prototype.
Test and review the prototype. Should be focused on how it works together and on the ecosystem.
Beta prototype
Polish
Implement the rest of features. Polish, balance and bug-fix, to create a final product.
Final product

Platform and Technologies

Target platform: Windows
Technologies and frameworks: Unity, Blender, Git
Programming language: C#

The original prototype was developed using the Unity game engine. We chose Unity because it is free for non-commercial purposes, its low skill entry barrier and its ability for rapid prototype development.

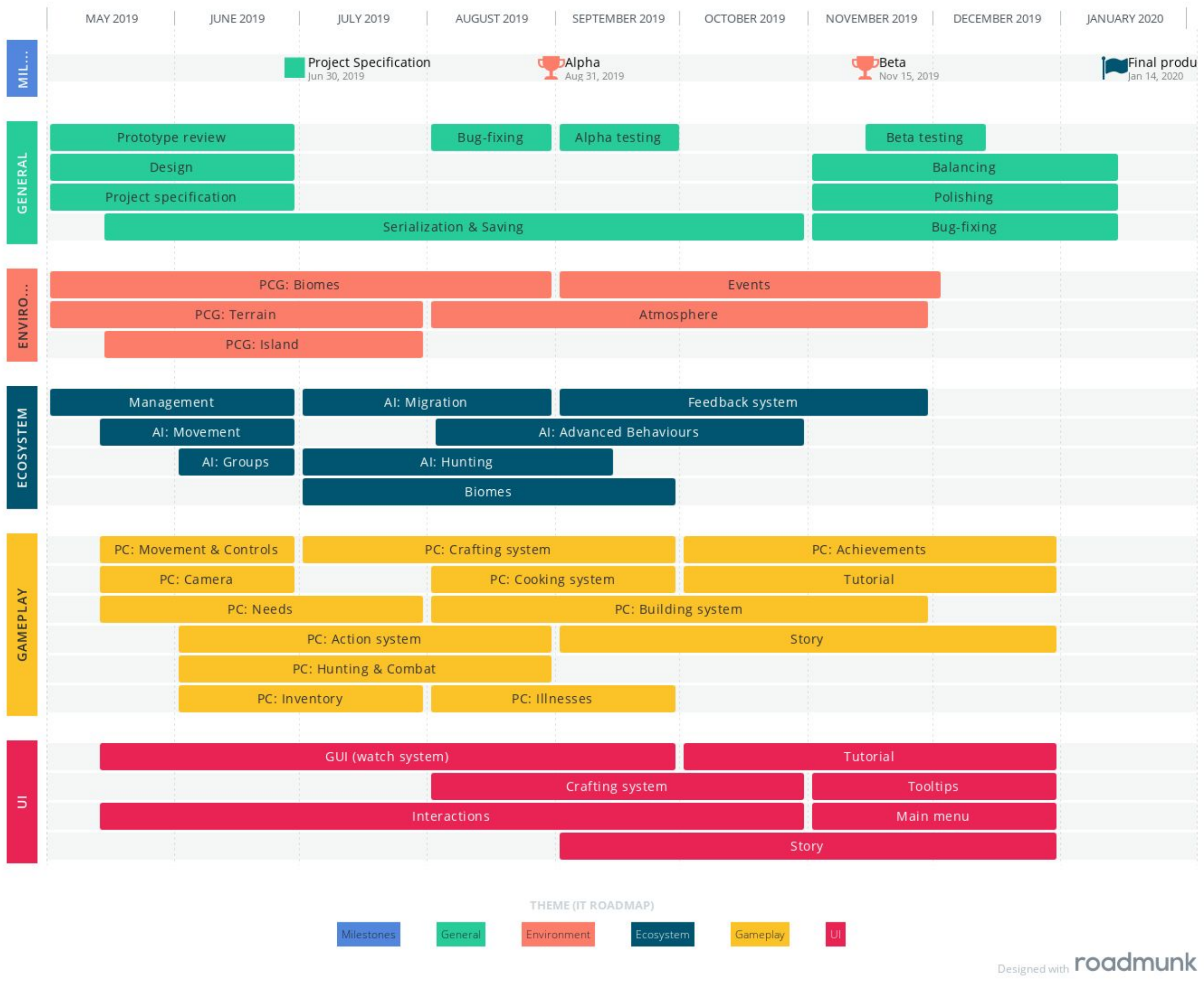
There are mainly two reasons we are once again choosing Unity for further development. The first reason is that we are already familiar with the development environment and workflow. The second reason is obvious: Although most of the code will have to be refactored and cleaned up, we avoid having to reimplement everything from scratch.

Unity is capable of deployment on over 25 different platforms. For us the main platform of both development and deployment is Windows. Other platforms for deployment that could be in consideration are macOs and gaming consoles (Xbox One, PlayStation 4).

We will be using Git for versioning both the game assets (sounds, 3D models, scenes, …) and the game code.

Estimations

In the project outline we have split the development into multiple phases. In the picture below we try to better split the phases and to allocate time for major features of the game.



Team roles

Given the experience obtained during development of the Domov prototype, we can assign roles as follows:

- **Jakub Svoboda**
  - Procedural generation of the environment
  - Creating the environment and its processes (mist, weather, ..)
- **Jan Kočur**
  - Programming the gameplay for the player (crafting, actions, ..)
  - Programming UI
- **Ekaterina Bessonova:**
  - Visual design
  - Creating art and animations for the game
- **Marius-Alexandru Anagnoste:**
  - Implementing ecosystem and its mechanisms
  - Quality assurance
- **Yulia Malkova**
  - Animal behaviours and movement

Project Scope

Discrete models and systems	
	Discrete mathematics and algorithms
	Geometry and mathematics structure in computer science
	Optimizations
Computer science	
	Computer science
Software and data engineering	
x	Software engineering
x	Software development
	Web engineering
	Database systems
	Analysis and processing of large data sets
Software systems	
	System programming
	Reliable systems
	Performance systems
Mathematical linguistics	
	Computer and formal linguistics
	Statistical methods and machine learning in computer linguistics
Artificial Intelligence	
x	Intelligent agents
	Machine learning
	Robotics

Computer graphics and computer games development	
x	Computer graphics
x	Computer games development