HØGSKOLEN I ÅLESUND

Aalesund University College

SWARM INTELLIGENCE

TUTORIAL

Harald Yndestad

Content

Foreword

This document is a teaching on the subject of Swarm intelligence. The course is divided into the following main topics are:

1. Individual Agents: How to manage individual agents.
2. Social Agents: How swarms of agents are collaborating in groups.
3. Evolution Agents: How swarms of agents are adapted to new landscapes.

The teaching program is built up as beginner's course with a sum of lessons. In each lesson is set up a set of learning outcomes, system theory, practice and exercise. Systems theory forms the basis for the theory of agent -based modeling. Procedure describes a typical method for modeling and programming of agents. Task describes exercise to meet the expected learning outcomes. Each topic has a procedure that describes how the theory of agent -based modeling may be used. We should start by developing a single framework for modeling, before it gradually introduced more complex models.

The document can be used in several ways. You can follow the plan from the start and then go through all the lessons. Eventually you can choose your own order or you can use some lessons as a starting point for your own research.

Harald Yndestad

1 INTRODUCTION

Question: What is Swarm?
Answer: A swarm is group of objects moving in a common direction. This swarm may represent people, eco systems, and a set of cars, ships, and more.

Question: What is Swarm intelligence?
Answer: Swarm intelligence means that there is cooperation between members in the swarm group.

Question: Why Swarm Intelligence modeling?
Answer: Traditional cybernetic methods are based on a population estimates. An estimate means that there is an error or an uncertainty in the estimate. This error has a source.

If you have a mutual relation between 3 ore more dynamic bodies, then we have a 3-Body problem and a complex system. Complex Systems have complex behavior and cannot be modulated by traditional mathematics. Den we need a different modulation approach. The Swarm intelligence approach is based on the idea of modeling complex system by modeling sets of individual objects.

Question: How do we modulate intelligent swarms?
Answer: Swarms are represented by a set of agents managed as a group.

Question: Is there an established method of Agent Modeling?
Answer: No. I do not think so. Agent Based Modeling is relatively new subject. It is not yet set up an established method for agent-based modeling, as we find in a find in object-oriented programming.

Question: What method is used in this teaching program?
Answer: In this teaching program we will use General systems theory as a framework for agent modeling.

Question: What is the advantage of this method?

Answer: It is a generic method that can be easily used by all types of agent -based modeling tasks.

2 AGENT SYSTEM MODELING

2.1 Introduction

Question: Is there an established method for modeling agents?
Answer: No. Agent Based Modeling is relatively new topic. It is not yet set up an established method for agent based modeling, such as we find in object-oriented programming.

Question: What method is used in this teaching program?
Answer: In this teaching program general systems theory has been chosen as baseline for a theory that have been developed by trial and error over years.

Question: What is the advantage of this method?
Answer: It is a generic method that can be easily used by all types of agent -based modeling tasks.

Learning goals
Students should at the end of this lesson:

1. Explain the concepts and methods for modeling agents based on general systems theory
2. Applying systems theory to agent-based modeling
3. Have a general knowledge of systems theory
4. Have a general knowledge of agent-based modeling

Students should at the End of this lesson could:

1. Explain the concepts and methods for modeling agents based on general systems theory
2. Applying systems theory two agent-based modeling
3. Have a general knowledge of systems theory
4. Have a general knowledge of agent-based modeling

2.2 General Systems Theory

A system S(t) is composed of a set partners P(t), collaborating on a common purpose, where

$$S(t)=\{B(t), P(t)\}$$

Where B(t) represents a set of relationships between a set of partners P(t). In this system each partner P(t) in S(t) represents a set of new subsystems. System elements S(t) have a set of dual model views.

$$S(\text{purpose}) = \{S(\text{otology}), S(\text{knowledge})\}$$
$$S(\text{ontology}) = \{S(\text{architecture}), S(\text{dynamics}, t)\}$$
$$S(\text{knowledge}, t) = \{S(\text{ethics}, t), S(\text{learning}, t)\}$$
$$S(\text{ethics}, t) = \{S(\text{purpose}, t), S(\text{restriction}, t)\}$$
$$S(\text{learning}, t) = \{S(\text{identification}, t), S(\text{control}, t)\}$$

System Models S(t) can represent all types that biological, social, technological and abstract organizations.

Agent based system model

Agents are related to landscapes. An agent-based model may represent by the model.

$$S(t)=\{N(t), A(t), L(t)\}$$

where A(t) represents set agents, L(t) is a landscape and N(t) is the relationships between agents, and between Agents and landscapes.

System Mutual Relation

There is a mutual relation between all system elements

$$A(t) = f(L(t), N(t)),\ L(t) = f(A(t), N(t)),\ N(t) = f(A(t), L(t))$$

This means that agents are affected by the landscape, and a landscape state is influenced by actions from agents. Agent network N(t) represents relationships between agents and between agents and landscapes.

Agent model views
Agents my be modulated by duals views:

S(agent, t)={A(Ontology, t), A(Epistemology, t)}

Where A(Ontology, t) represents an external view of the agent properties and A(Epistemology, t) represents the internal agent knowledge. These properties may be modulated a new set of a new set of dual views

S(Ontology, t)={A(Architecture, t), A(Dynamic, t)}
S(Epistemology, t)={A(Ethic, t), A(Learning, t)}

where A(Architecture, t) represents the agent's architecture, A(Dynamics, t) represents the agent dynamics, A(Ethic, t) the agent ethics goals and A(Learning, t) the agent learning methods.

Agent Architecture
Agents have an architecture composed by a set of agent services:

A(Architecture, t) = {Sensor, Body, Producer, Motor, Control, Output}

Where Agent(Sensor, t) represents a part of the agent network that measures conditions in landscape and to other agents. Agent(Body,t) represents the properties of the agent's form. It includes visual properties, friction against the surroundings and the like. Agent(Producer,t) is a service that performs transformations to consume anything and produce something. It can be a product, a service, agent resources, or the agent recruitment. Agent(Motor, t) is a service that translates agent resources to a movement in the landscape. Agent(Control,t) performs control functions in the agent. Agent(Output,t) is part of the agent network that may affect the state of the landscape or other agents.

Agent Dynamics
Agent dynamics has an internal and an external reference. The internal dynamics agent has properties

A(Dynamics, Internal, t) = {State, Behavior, Goal }

Agent(Dyn, State, t) represents the agent's state Dynamics. Examples of state dynamics are dynamic properties compared to the movement, growth, production and the like. Agent(Dyn, Behavior, t) represents the agent's behavior dynamics. The term behavior is related to the sequence or phases of operations to achieve goals. Agent(Dyn, Goals, t) represents changes in the agent's goals. Agents are homing objects. Once the target also has a dynamic, meaning there that agents also have the ability to choose new targets.

Agent Ethics
All agents are goal seeking by nature. Agent ethics has characteristics

A(Ethic, t) = {Goal, Limitations}

At the same time so that objectives related to the agent's limitations. Agent ethics A(Ethic, t) has an internal and an external perspective. Agent internal ethics A(Ethic, Internal., t) seeks to optimize the agent's own resources. Agent external ethics A(Ethic, External, t) seeks to adapt the agent's goals and abilities for a purpose with external reference. All agents represent a system element.

Agent Learning
Agent Learning A(Learn, t) is to adapt the agent states so that realizes the agent 's internal ethics A(Ethic, Internal, t) and external ethics A(Ethic, External, t).

Internal Learning A(Lea, Internal, t) is learning to optimize its resources. External learning is optimizing costs to external conditions. It may be identifying targets with resources, and to achieve the goals with the least possible cost.

2.3 Individual Agents

Agent internal properties can be modeled with the system perspectives

S(Agent ,t) = {A(Arc,t), A(Dyn,t), A(Eti, t), A(Lea,t) }

where A(Arc,t) represents the agent's architecture, A(Dyn,t) agent dynamics , A(Eti,t) agent ethics and A(Lea, t) agent learning.

Agent architecture A(Resources, t)

Agents have an internal architecture, which is composed of a set of services :

A(resources) = {Sensor, Body, Producer, Motor, Control, Output}

Agent(Sensor,t) represents a part of the agent network that measures conditions in landscape and to other agents. Agent(Body,t) represents the properties of the agent's form. It includes visual properties, friction against the surroundings and the like.

Agent (Producer,t) is a service that performs transformations to consume anything and produce something. It can be a product, a service, agent resources, or the agent recruitment. Agent(Motor,t) are services that translates agent resources to a movement in the landscape.

Agent(Control, t) performs control functions in the agent.

Agent(Output, t) is part of the agent network that may affect the state of the landscape or other agents.

Agent dynamics

Agent dynamics has an internal and an external reference. The internal dynamics agent has properties

A(Dyn, Internal , t) = {State, Behavior, Size}

Agent (dyn, condition, t) represents the agent's state Dynamics. Examples of state dynamics are dynamic properties compared to the movement , growth, production and as like.

Agent(Dyn, Behavior, t) represents the agent's behavior dynamics. The term behavior is related to the sequence or phases of operations to achieve goals.

Agent(Dyn, Goals, t) represents changes in the agent's goals. Agents are homing objects. Once the target also has a dynamic, meaning there that agents also have the ability to choose new targets.

Agent ethics

Agent business ethics has characteristics

A(Ethics, t) = {Dimensions, Limitations }

Agents are target seeking by nature. At the same time so that objectives related to the agent's limitations.

Agent ethics A(Ethics, t) has an internal and an external perspective. Agent internal ethics A(Ethics, Intternal, t) seeks to optimize the agent's own resources . Agent external ethics A(Ethics, External, t) seeks to adapt the agent's goals and abilities for a purpose with external reference.

Agent learning

Agent Learning A(Learn, t) is to adapt the agent states so that realizes the agent 's internal ethics A(Ethics, Internal, t) and external ethics A(Ethics , External, t).

Internal learning A(Read,int, t), the agent learns to optimize their own resources. External learning lea, the agent learns to optimize costs in relation to external conditions. It may be identifying targets with resources, and to achieve the goals with the least possible cost.

Performance
Intelligent agents can change properties by monitoring the agent 's own performance. A method for monitoring the agent 's performance is to calculate a cost function.

J(t) = average(e(t)*Qe(t), u(t)*Ru(t))

Where e(t) represents deviations from targets, Q represents the cost of the deviation, u(t) represents an external control signal and R represents the cost of the external control power.

2.4 Social Agents

Social agents are a set of agents with common rules. Common rules means that social agents represent a common system.

Social Agent architectures
A set of social agents together forms an Social Agent Architecture. The social architecture says something about how agents are connected to each other via the network N(t) . Typical examples of Social agent architectures are:

A(Social, Architecture, t) = {Group, Swarm, Queue,,,, }

Social Agent dynamics
Social agents dynamics says something about how the group evolves over time. Typical examples of Social agent dynamics are:

A(Social, Dynamics, t) = {Group velocity, Group speed direction}

Social Agents ethics
Social agents ethics says something about the group motive to collaborate on a common purpose. Typical examples of Social agents ethics are:

A(Social, Ethics, t) = {Target, Target value, Fitness, Risk }

Where Target is the target where an agent will find a resource, Target value represents a target priority, Fitness an adoption to a wanted state, Risk is related to an unwanted cost.

Social Agent Learning

Social agents Learning says something about the group rules to realize the social ethics:

A(Social, Learning, t) = {Internal learning rules, External Learning Rules}

Typical internal rules are genes, control parameter, control rules and cost functions. Typical external rules are fitness functions and cost functions.

Performance by cost functions

Intelligent agents can change properties by monitoring the agent's own performance. A method for monitoring the agent 's performance is to calculate a cost function.

J(t) = average(e(t)Qe(t), u(t)Ru(t))

Where e(t) represents deviations from targets, Q represents the cost of the deviation, u(t) represents an external control signal and R represents the cost of the external control power.

Social Agent architectures

A set of social agents forms a Social agent architecture. The social architecture says something about how agents are connected to each other via the network N(t). Typical examples of Social agent architectures are:

A(Social, Arc, t) = {Group, Swarm, Queue}

Social Agent dynamics

Social agent dynamics is how agents are changing in time. Typical examples are::

A(Social,Dyn, t) = {Group speed, Alignment direction}

Social Agents ethics
Social agents ethics says something about the group motive to collaborate on a common purpose. Typical examples of Social agents ethics are:

A(Social, Ethics, t) = {Identify resources, Grazing resources, Reduce risk}

Social Agent Learning
Social agents Learning says something about the group rules to realize the social ethics. Typical examples of Social agents learning is the identification and control of:

A(Social, t) = {Threat, Resources access, Risk}

2.5 Procedure

An Agent System Model may be modulated by the following procedure.

1. Crete a System model
Create a Agent Based System Model. Start with a short description system name of S(t) the system purpose, name of agents A(t), the name of the landscape L(t), and describe typical relations N(t) between agents and landscape.

2. Create an Agent model

1. Create an agent model A(Architecture, t). Describe the service functions in the agent architecture.

2. Formulate the agent ethic A(Ethic, t). Set agent targets, values and possible fitness functions.
3. Describe the possible agent dynamics A(Dynamics, t), the important state dynamics and flow dynamics
4. Describe the agent learning strategies A(Learning, t) the will let agents go for the agents ethic A(Ethic, t).

3. Create a Social Agent Model

Social agents have a set of common rules

1. Describe the purpose or ethics of the social agents
2. Formulate the common social rules
3. Formulate how the social agents will learn to follow the common rules

2.6 EXERCISE

Exercise 1: About System modeling

1. Do you know for the definition of a system model?
2. Describe the system properties of an agent based system model
3. Describe an individual agent system model
4. Describe a social agent system model
5. What is a swarm?
6. What is an intelligent swarm

Exercise 2: Producer and Consumer system

Modulate an agent-based system where the system has a set of Producers, Consumers and moveable transporters between Producers and Consumers.

Control Questions about agent -based systems

About knowledge

1. Do you guys for the definition of a system model
2. Describe the system properties of an agent -based system model

About skills

There should be developed an agent -based system model for industrial farming

1. Formulate the characteristics of a typical landscape model
2. Identify the typical agent groups in the model
3. Formulate the model purpose and ethics
4. Formulate typical rules for learning

2.7 Literature

1. Ola Larses and Jad El-khoury: Views on Systems Theory
2. Yndestad H. General Systems Theory

3 INDIVIDUAL AGENTS

3.1 Starting agent programing

Question: What is an agent?
Answer: An agent is a goal-seeking object.

Question: What is the difference between object and agents?
Answer: The difference is that objects transformations data through algorithms. Agents are goal oriented objects.

Question: Is not this the same?
Answer: No. Not quite. Objects are based on a deterministic paradigm and can be compared with machines that produce predictable results. Agents are based on a statistical paradigm with free will to optimize a service.

Question: Is there a standard method for modeling agents?
Answer: No. Not really. Agent based modeling can be based on many concepts. Attempts are made to develop standard software for agent-based modeling. Meanwhile we see that this is tied to particular applications.

Question: What about a standard framework?
Answer: When programming agents might be fine to start with a standard framework for modeling and programming of agents. In this teaching program, I have chosen general system theory for modeling agents. This concept is chosen in order to use a generic theory of agent modeling. On this basis, we shall first make a framework, and then develop further models step by step, until an agent that performs more complex tasks. In addition, it may be okay to use a standard scripting framework that is used as a template for programming agents.

Question: How can we create a framework for agent programming?
Answer: A straightforward method is to start with a framework for real-time mechanisms to the agents. Once this is in place it may be

okay to get established an agent server that creates and manages a group of agents. Agent services can then develop gradually as needed. It is then customary to start with a hello Agent.

Learning objectives

A discontinues this lesson.

Knowledge

1. Be able to explain the difference between objects and agents

Skills

1. Could create a Hello Agent as a framework
2. Could create visual Agents
3. Could create a 3D landscape Unity game engine
4. Be able to import terrain model to Unity game engine
5. Could use basic services in Unity game engine
6. Could create a framework for an agent-based modeling

General knowledge

1. If the agent framework
2. About Unity game machine as technological platform

Topics

1. Agent framework
2. Hello agent and visual agents
3. Real-time Control of agent state
4. Reporting of agent states
5. Textures, sea, sun, camera and background landscape
6. Unity as technological platform.

Technological Platform

Before you begin programing agents, you should think through what technological platform and programming language to use.

Question: Can a use Matlab programming of agents?
Answer: The answer is yes. In Matlab can represent landscapes and agents in tables. The landscape is modeled as a 2D array. Agent properties can be stored in a vector, which is stored in a list. Do you latest version of Matlab, can be programmed agents as objects and stored in lists. The advantage of using Matlab, is that you have direct access to other services such as Matlab GA, Neural network, etc.. It can often be helpful to start with Matlab when you try out an algorithm. The downside is that Matlab lacks visual representation of agents and landscapes, and many of the real-time services found in a game engine.

Question: Can a use Java for programming of agents?
Answer: The answer is yes. Java is an object-oriented language. This significantly simplifies the programming work. In Java, the landscape is represented as a 2D array. Agents programmed as objects and stored in lists. The advantage of using Java, may be that you have more knowledge about this language. The disadvantage is that Java lacks the visuals and real-time services found in a game engine.

Question: Why use Unity game engine?
Answer: Unity is used by more than 250,000 game programmers and is considered one of the best games engines on the market. Landscape programmed as 2D array. Agents are programmed as visual agents and stored in lists. The advantage of using Unity game engine, is that a visual presentation of the agents. In addition, the game engines a well-developed framework for control of real-time processes and the control of interaction between agents. This allows Unity is particularly suitable for modeling agents, games and modeling of complex systems. The disadvantage of Unity is that one must put into a technological platform, which in reality is a development system.

Unity can be programmed in multiple languages. The main is JavaScript and C#. JavaScript is a simple programming language.

The most Tutorials therefore uses JavaScript in their examples. The disadvantage is the lack of standardization. In this document I have the choice to use C #, since this is closer to Java program code.

Question: Do we need the Unity game engine?
Answer: Unity is an integrated development system. It's easy to get started to get to something that works and it's easy to create a complex landscape. At the same time large development system with room for the development of complex systems.

A good approach is that you:

1. Starting with downloading a free 30-day version. See some of Iceland Demo. They get a sense that the possibilities inherent in Unity as development tools.
2. Take a review of video tutorials. Then you see how you can build up a scene.
3. 3. Build up some simple scenes even over a few lessons. Then you get programming into fingers. Try eg Fireball and "Hello World in 6 minutes". These contain the most you need.
4. Look especially at script programming in Unity.
5. When you start programming agents, it may be wise to start with a blank scene. Agents visualized as a bullet, and landscape as a surface.
6. When things start working, you can gradually put on more complex 3D visual scenes and agents, who will grace the landscape.

Unity information

Question: Where can I find information about Unity?
Answer: If you start here, you will find most.

1. Unity http://unity3d.com
2. A Unity free 30 day trial here: http://unity3d.com/unity/download.

3. Unify Community Wiki (UCW) http://www.unifycommunity.com/wiki
4. Unity Forums: http://forum.unity3d.com
5. Unity Documentation: http://unity3d.com/support/documentation
6. Virtual More Wiki: http://virtualmore.org/wiki
7. Virtual More Blog: http://www.virtualmore.org/blog/

Unity demos

Question: Can I demo videos about using Unity online?
Answer: Yes. Here is a good beginning

1. YouTube: http://www.youtube.com/watch?v=q3Cy0lAd6gk
2. Unity demos: http://unity3d.com/support/documentation/video/

Unity tutorials

Question: Is it enough to study some videos?
Answer: No. Unfortunately. You need enough start at the beginning with practical exercises, so you get it into your fingers. You can then start her

1. Unity tutorials: http://unity3d.com/support/documentation/video/
2. Championships in 6 minutes: http://www.youtube.com/watch?v=LP73jeWtaNY
3. Fireball: Note Robin Trulssen Bye.
4. Virtual More Tutorials: http://virtualmore.org/wiki/index.php?title=Tutorials

Video examples of particle agents

1. Particles in a river: http://www.youtube.com/watch?v=qkZGcNyrlOk
2. Particle in a river: http://www.youtube.com/watch?v=iW0C4M4yY5U&feature=related

3. Massive particles: http://www.youtube.com/watch?v=7cjorOe810o&feature=related

Resources

1. Unity game engine: 30 day free trial is available here: http://unity3d.com/unity/download.
2. Access to a 3D model of an object. Certainly developed in 3D tool Blender, Maya or 3D Studio. Models of this type are readily available via the Internet.
3. Access to a 3D model of a map
4. Download Unity 3.2: http://unity3d.com/unity/download/
5. Introduction to Unity: http://unity3d.com/support/documentation/video/
6. Wiki Unity resources: http://www.unifycommunity.com/wiki/index.php?title=Main_Page
7. Unity Web forum: http://forum.unity3d.com/viewtopic.php?t=34451
8. Unity Tutorials: http://www.reddit.com/r/unity_tutorials/
9. Unity Lab: http://www.unitylabs.net
10. Unity Studios: http://www.youtube.com/user/UnityStudiosCOM#p/u/0/AdsKJFU

Virtual More Resources:

1. VM Blogg: http://www.virtualmore.org/blog/
2. VM Wiki: http://virtualmore.org/wiki/index.php?title=Main_Page
3. VM hjemmeside: http://www.vrmore.no/
4. VM Demo: http://web.me.com/virtual_more/Websted/Velkommen.html
5. Ship simulation: http://www.youtube.com/watch?v=q3Cy0IAd6gk
6. Hello World: How to make a Terrain in 6 minutes: http://www.youtube.com/watch?v=LP73jeWtaNY

Some YouTube Videos

1. http://www.youtube.com/watch?v=5wxe1IUu5QA
2. http://www.youtube.com/watch?v=dm5vEzzv1wY
3. http://www.youtube.com/watch?v=RkglrB6CBrE
4. http://www.youtube.com/watch?v=GKc99RO9VTI
5. http://www.youtube.com/watch?v=1aiMP1sAFd8
6. http://www.youtube.com/watch?v=WeTP04Gi5CY
7. http://www.youtube.com/watch?v=hoeuwAnqWF0
8. http://www.youtube.com/watch?v=Hr0ZI142P5M
9. http://www.youtube.com/watch?v=iwUkJgbSb0c
10. http://www.youtube.com/watch?v=cwwcDkHgkCk
11. http://www.youtube.com/watch?v=wiZC5n1821g

Literature

Question: Is there textbooks on Unity?
Answer: Yes. It is now a series eBooks about Unity on the market.
See: www.PacktPub.com

1. Will Goldstone: Unity Game Development Essentials. PacktPub

3.2 Hallo Agent

Question: What is a hello Agent?
Answer: Hello Agent is your first agent. This is a good beginning. Service to Hello Agents is reporting a simple hello message condition to a user. This is the first step in the development and testing of the agent properties

Question: How can we create a simple Hello Agent?
Answer: The easiest way is to create method ReadMe(). This method allows you to insert in Agent script. With simple changes can this simple method used later for new tasks. Eventually, you will also use other methods to communicate with the agent. When it gets many data to keep track of, you can read the global variables in Unity inspector, or can create a GUI that read conditions in real time.

LEARNING OBJECTIVES

A discontinues this lesson the following

Knowledge
1. Could create a hello agent who report their own condition
2. Could create a landscape on a game engine skills
1. Could use basic services in Unity game machine
2. Could create a framework for an agent-based modeling

General knowledge
1. If the agent framework
2. About Unity game machine as technological platform

Topics

1. Reporting of agent state
2. Real-time Control of agent state

SYSTEM THEORY

Hello Agent has a system architecture with only one service

A(arc, t) = {Output}

where Output provides a simple survey to nearby destinations. In this case, it is an object that reports out a single message to the user. This simple service is a good start to developing a more full-fledged agent.

Procedure

Modeling

1. Set up a simple system model for agent
2. Formulate agent methods for performing service.

Programming

1. Start a new project in Unity
2. Create a new directory "myScenes" under "Project". Select "File" "Save Scene", giving the stage name Hello Agent Scene and move file to "myScenes.
3. Create a C # -file under the "Project" and enter the enclosed script for "Hello Agent". Write about file name to "Hello Agent" and add the file under a new directory "MyScript". See attached script example-
4. Select "Game Object" "Create Other" "Sphere". "Sphere" under "Hierarchy" rewritten "Agent Object".
5. Drag the script "Hello Agent" to "Agent Object" under "Hierarchy".
6. Select "Hello Agent Scene" and start the program via the Arrows. The message from the agent is printed on the bottom left.

Exercise

Control Questions about intelligent agents

1. What kind of services has a Hello Agent?
2. What framework are related to the programming of a hello Agent
3. How can an Hello Agent are further used for programming of agents?

Experiments with Hello agents

1. Create a Hello Agent as described above. Modify the agent's program code so that it prints the agent numbers every second
2. Create a new method newNumber () that inserts a random agent number.

Verification

1. Test whether the agent exerts the desired service

3.3 Visuell agent

Question: What is a visual agent?
Answer: A visual agent is an agent with 3D representation of the agent's visual characteristics.

Question: What is the usefulness of visual agents?
Answer: Usefulness of visual agents is to produce a visual representation that represents the agent. By modeling simple systems can introduce visual agents easily be extra hassle and annoyance. It is then easier to portray agents as small spheres. When you develop more complex scenes you will like to have a better overview of the agents represent. A visual agent representing the agent's properties can then be a good idea and. This is common in computer-based games.

Question: How can we bring up a good visual representation of an agent?
Answer: associating a visual 3D model directly to the agent in the game engine can do it.

Learning outcome

A discontinues this lesson.

Knowledge
1. Be able to explain the use of visual agents

Skills
1. Could create visual Agents
2. Could use basic services in Unity game machine
3. Could create a framework for an agent-based modeling

General knowledge
1. If visual agents in games and 3D modeling.
2. About Unity game machine as technological platform.

Topics

1. Visual announcer agent
2. Real-time Control of agent state
3. Reporting of agent state

Systems Theory

Visual agents have a body that represents the agent's visual design. In this case, the agent no other functions. It can be formulated with the simple system architecture

$$A(arc, t) = \{Body\}$$

where Body represents the agent's visual characteristics of the landscape.

PROCEDURE

Programming

1. Creation of 3D object: Create a visual 3D model of an object in a 3D modeling tool. Using the Maya, Blender, or 3D Studio, the object directly imported into Unity.
2. Import of 3D object: Select Assets and Import New Asset. Then you go into a directory and select the 3D object, and load 3D object. The object is now present as a prefab object in the project directory.
3. catalog 3D objects: Creates a directory Resources under Project, and move the object into the subdirectory Resources.
4. Scale: Drag the visual object into the scene and adjusts the desired scaling, surface and similar under the "Inspector".
5. Visual Agent: Drag the imported 3D models into Agent Object under Hierarchy. You will now see the 3D object in the scene and in the Game window. You can now make copies of the agent by dragging new copies of prefab into the scene.

Verification

1. Check if the print is as expected
2. Insert reports in agent if there are deviations from what you expected

EXERCISE

Control Questions about visual agents

1. What is a Visual Agent
2. Why it might be useful that agents have a visual representation?
3. Where is the visual agents preferably being used

Experiments with visual agents
In this task, we can start to develop a simple visual 3D representation.

1. Modeling: Choose a finished 3D model of an object developed with a 3D modeling tool that Blender, 3D Studio or Maya.
2. Programming: Create a visual agent as described above.
3. Discussion: Will copies of visual agents produce the same agent number?

3.4 3D Landscape

Question: What is a 3D landscape?
Answer: A 3D landscape is the same as a terrain model. Landscape is sedentary information outside agent. Agents can normally read information landscape and simultaneously affect information landscape. It is then so the agent must have access to an adapted landscape, while it must be able to characterize the landscape.

Question: What benefit has use landscape?
Answer: Modeling of landscape is a method to organize sedentary information that communicates with agents. When landscape gets a visual representation, it is easy to identify complex relationships.

Question: How can we create a 3D landscape as a terrain model?
Answer: You can create a landscape as a 3D object in Unity. This can then be designed manually as a terrain model and then applied texture, so it looks pretty.

LEARNING OBJECTIVES

A discontinues this lesson

Knowledge

1. Be able to explain methods for developing 3D landscape on game engine

Skills

1. Could put texture, sea and background to a landscape
2. Could create manually a 3D landscape on a game engine
3. Be able to import terrain model
4. Could decorate the landscape to a more realistic visual scene

General knowledge

1. Could use basic services in Unity game machine
2. Be familiar with the opportunities to develop 3D landscape in game machines

TOPICS

1. Manual modeling of terrain
2. About a 3D terrain model
3. Texture, vegetation, sea, sun, camera and background

SYSTEMS THEORY

Agents in landscape is part of the meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

Where L (t) represents the landscape, A(t) agent and N(t) the agent's relationship to the landscape.

PROCEDURE

Manual creation of 3D terrain model

1. Create a terrain: Select Terrain and Create Terrain
2. Adjust terrain: Adjust terrain angle to camera.
3. Set height: Select Raise High the Inspector and well an appropriate scaling to change map. Let arrow "paint" in place the terrain you want. Turn so the terrain to a natural angle

Imports of elevation map

1. Height Map: You are based in a finished height map of DEM format.
2. Import: You import elevation map from a catalog by selecting Terrain and

1. Create Terrain and Terrain again and Import High map. Then select the saved map file.
2. Scale: In windows Import high map select the platform is Mac or PC. Select Terrain Size and set scaling to map (Ålesund: 15500x, 1100y, 15500z). Then select import.

Textures terrain model

1. Selecting texture: Select Terrain object. Choose a paint brush under inspector. Then select Edit and Add texture. You'll then see a list of possible textures.
2. Type: Select scaling the texture that matches height resolution (eg 1550x15500m), and then Add.

Vegetation terrain model

1. Selecting texture: Select Terrain object. Choose a paint brush under inspector. Then select Edit and Add grass. You'll then see a list of possible grass or vegetation types.
2. Type of vegetation: Select Type grass, wood or plant.

3. Composition of vegetation: Vegetation can then "painted" on the terrain with brush.

Sea Surface terrain model

1. Choose water texture Go into the Projects directory under Default assets. Here is prefab Daylight water.
2. Scale: Drag Prefab Daylight water into terrain. Adjust as height, position and scaling of prefab Daylight water.

Sky Background terrain model

1. Choose texture: Choose Edit and Render Settings. Select Blue Sky in the Inspector.
2. Show mode: Press the Scene iconic under Scene upper left.

Light terrain model

1. Select source: Select Game Object. Create Other and Direct mortality. You will then forward a sun
2. Orientation: Scale sun's position and direction.
3. Light map: After the sun has found a natural position create standard light map allowing sun and shadows inserted into the terrain. You do this by selecting Terrain and Create light map, Mark shade if you want this with.

EXERCIZE

Control Questions about agents in landscape

1. What is meant by an Agent landscape
2. What is the agent-based landscape can represent?
3. How is the link between agents and landscape?

Experiments with agents in landscape

In this task, the development of a visual 3D landscape Unity game engine.

1. Study the video Championships in 6 minutes: http://www.youtube.com/watch?v=LP73jeWtaNY
2. Terrain Model: Import terrain model for Møre

3. Texture: Goodbye terrain, texture, sea, sun and background
4. Discussion: What do you think are the possibilities and limitations of 3D modeling of landscape?

3.5 Passive Particle agents

Question: What is a passive particle agent?
Answer: A passive particulate agent has no free will to adapt a remote environment.

Question: What are examples of passive particle agents?
A: Examples of passive agents are particles that represent objects or materials in free fall, or running along a landscape.

Q: What can passive particle agents in stationary landscape do?
Answer: Typical tasks are to identify complex flow patterns. Flow pattern in the water, waves etc.

LEARNING OBJECTIVES

Students should at the end of this lesson could:

1. Develop a system model for particle agents
2. Planning an experiment using particle agents
3. Could simulate a passive particle agent in free fall
4. Create a passive particle swarm agents

SYSTEMS THEORY

Meta system for a simple agent in a landscape can be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents an agent, L(t) landscape and N(t) is the relationships between agent and landscape.

The network has a relationship N(t) = {A(t),L(t)} where A(t) represents an agent with stationary link between agent and landscape.

We imagine that the landscape L (t) is represented with a 3D matrix in which agents A(1,t) consume resources from cell L(1,x,y) and produces something to 2D landscape in cell L(2,x,y).

In this case, the landscape is constant. It can be formulated as:

$$L(x, y, z, t) = K(x, y, z)$$

where K(x, y, z) is a 3D representation that does not change with time.

Agent Model
Agents can be modeled with the properties

$$A(t) = \{A(Arc, t), A(Dyn, t), A(Ethics, t), L(Learn, t)\}$$

Where A(Arc,t) represents the agent's architecture, A(Dyn,t) Agent state dynamics, A(Ethics, t) agent ethics, and L (Learn, t) agent learning strategy.

Agent architecture A(arc, t)
Typical agent architecture A(Arc, t) for a place tied agent will be:

$$A(Arc, t) = \{Sensor, Body, Motor\}$$

Where sensor measures and updates the state of the landscape L(z, x, y, t), Body represents the agent form a visual agent. Engine is a service that maintains the agent's progress. Typical services for a particle agent will be:

A(Sensor, t) = {reader agent close environment}.
A(Body, t) = {Represents particle geometry and friction}

A(Motor, t) = {Safeguards particle progress}

Agent dynamics A(Dyn, t)
Agent dynamics represents the agent's dynamic properties of how it evolves in time. A typical dynamic model for a particle model will be:

Particle acceleration vector: a(Acc., t+T) = A(Acc, t) + F(Engine, t)/Mass
Particle force vector: F(Engine, t) = -K(Friction) v(Part, t) + F(External, t)
Particle velocity vector: v(Part, t + T) = v(Part, t) + T*A(Part, t)
Particle position vector: x(Part, t + T) = x(Part, t) + T*v (Part, t)

Where T is the sampling interval.

Particle Agent ethics and learning
Particle Agents have no free will and therefore has no systemic ethics. General Agent learning is the realization of the agent's ethics. When agents have no ethics, they have no ability to learn.

PROCEDURE

Agent Based Simulation of passive particle model can be considered as a laboratory experiment where the basis of the following procedure.

1. Set up a meta system model experiment
2. Set up an affiliate model
3. Formulate agent purpose and methods
4. Applications agent properties
5. Documents the agent properties
6. Discuss your results

Procedure for using Unity physics engine:

1. Create a stationary landscape that captures falling agents
2. Assume Hello Agent and create a new particle agent
3. Introduce gravity for the agent in the agent's physics engine

4. Introduce Collision box around Agent
5. Study how a single agent falls in the landscape
6. Create an Agent Server for creating a group agents
7. Create a group agents where you test out the motion of an overall group particle agents

Procedure for the application of particle engine:

1. Turn off the gravity Unity physics engine
2. Create a method Agent Motor () that affects the agent calculates the agent velocity vector
3. Introduce gradually new services in Agent Motor (). First try a simple agent where you introduce a gravity. Then introduce you friction to study what happens.
4. Create an Agent Server for creating a group agents
5. Create a group agents where you test out the motion of an overall group particle agents

EXERCISE

Control Questions about particle agents

1. Explain what a passive particle agent means.
2. What can a passive particle agent represent?
3. How can we create a physics model for a passive particle agent?

Experiments with particle agents

In this experiment we will study how a group of passive particulate agents, with mass, behave in an enclosed landscape.

1. Create a random closed landscape and study how a group of passive particle agents behave when agents use gravity from Unity physics engine.
2. Create a random closed landscape and study how a group of passive particle agents behave when agents use their own physics model

3.6 ProCumer Agents

Question: What is a ProCumer Agent?
Answer: ProCumer Agents are consuming resources, producing resources, and transforming resource to a new state.

Question: What are examples of ProCumer Agents?
Answer: Typical examples of ProCumer Agents are agents who will represent industrial or biological systems. Some examples of industrial systems, cars, ships, robots and production systems. Some biological examples are agents representing plants, trees, animals, people and the like.

Question: What can ProCumer Agents do?
Answer: Typical tasks are solving optimization problem. It may be optimal resource utilization, production capacity, local risk, timing for optimal growth, distribution of vegetation, distribution of market etc.

Question: How can we model ProCumer Agents?
Answer: ProCumer agents can be modeled by using a resource from a landscape or from other agents and then produce new resources. Manufacturer agents and biological agents transforming like a material from one form to another. Agents representing cars, ships or robots, transforming like a resource into a form of motion. Agent optimization problem here is to adapt its cost and production capacity in relation to available resources.

Question: How can an agent choose a resource?
Answer: Manufacturer agent can select a resource by putting a value on the resource, and a charge of the execution of the transformation. The agent selects resources for a cost / benefit assessment.

LEARNING OBJECTIVES

Students should at the end of this lesson could:

1. Explain apply reads Manufacturer Agents
2. Modeling a Manufacturer Agent

3. Perform an experiment with the use of Producer Agents

SYSTEMS THEORY

Meta system for a simple agent in a landscape can be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents an agent, L (t) landscape and N(t) is the relationships between agent and landscape. The network has a relationship N(t) = {A(t), L(t)} where A(t) represents an agent with stationary link between agent and landscape.

Landscape L(t)

The condition of consumer agents connected with the relationship between a production process in the landscape L(t) and a consumer process agent A(t). This leads to a feedback process in which

$$L(t + T) = f(A(t) + L(own, t)) \text{ and}$$
$$A(t + T) = f(L(t))$$

per T is a sampling interval and L (own, t) represents a separate production in the countryside. A typical growth model can be modeled as a logistic model. This can be formulated as:

$$L(t +T) = (1 + T*a) L(t) - (T*b) L(t)*L(t)$$

where a represents the growth rate and b represents the reduction rate in the countryside.

Agent Model A(t)

Agents can be modeled with the properties

$$A(t) = \{A(arc, t), A(dyn, t), A(eti, t), L(leather, t)\}$$

where A(arc, t) represents the agent's architecture, A(dyn, t) Agent state dynamics, A(eti, t) agent ethics, and L(leather, t) agent learning strategy.

Agent architecture A(arc, t)
A typical agent architecture A(arc, t) for a place tied agent will be:

A(sheets, t) = {Sensor, Body, Producer, Controller}

which sensor measures and updates the state of the landscape L(z, x, y, t), Body represents the agent form a visual agent. Producer transforming consumed resources into a new state or a new resource. Make exercising a control for learning agent ethics. In this case one can imagine that the agent has the following services:

A(Sensor, t) = {Reading state in the landscape}
A(Body, t) = {A simple geometry}
A(Producer, t) = {A transformation from a consumer to a product}
A Controller, t) = {Control of consumer}

Agent ethics A(eti, t)
Agent ethics has characteristics A(eti, t) = {goals, constraints}. That agent seeks to realize goals within the framework of the agent's limitations. Typical properties here will be A(eti, t) = {optimal consumption over time, capacity}.

In this case one can imagine that the agent's ethics are:

A(Ethics, t) = {To optimize their own consumption in landscape over time}

Agent consumption in a landscape forms the basis for the agent's production, which in turn is associated with a cost.

Agent Learning A(leather, t)

Agent learning is to exercise the control needed for realizing agent ethics. This learning has one perspective related to the environment and one related to the agent. This can be summarized with the characteristics A(leather, t) = {Identification, Control}. That is an identification of landscape condition and a control of the agent's state. Landscape condition identified by the agent sensor, via the agent network.

Learning can be exercised in accordance with various control strategies. One simple method is to exercise control in accordance to the control lav.

$$U(t) = [R-L(t)]\, K(t), \text{ for } U(t) > 0$$

where R is the desired level on the landscape, L(t) is identified state in the landscape condition L(x, y, t) and K is the agent's emphasis on deviation

Optimal learning
Optimal learning is then to optimize the cost function

$$J = (\text{Consumer value, Cost})$$

That agent's choice of resources in the landscape, must be related to the cost associated with retrieving the resource.

PROCEDURE

The application of Transform Agents can be considered as a laboratory experiment where the basis of the following procedure.

1. Set up a meta system model experiment
2. Set up an affiliate model
3. Formulate agent purpose and methods
4. Applications agent properties
5. Documents the agent properties
6. Discuss your results

Procedure for method development

1. Formulated physical properties of the landscape L (t). If the landscape produces a resource, formulates a model for their own development of landscape dynamic development
2. Formulate the resource value of the landscape
3. Formulate agent cost by consuming a resource in landscapes and transform the resource on a new shape.
4. Formulate agent costs.
5. Formulate a strategy for how much of the resource agent will consume

EXERCIZE

Control Questions about production agents

1. Explain what is meant by Production Agents,
2. What can Production agents represent?
3. How can a production agent customize his or her own production capacity resources in a landscape?

Experiments with production agents

In this we will study how a simple Production agent can customize his or her own production capacity to a resource that is produced in a landscape. We imagine then that landscape produces a resource in accordance with a logistic model

$$L(t+1) = a*L(t) - b*L(t)*L(t)$$

Where a and b are selected constants.

Production agent chooses an amount of scenery L(t), while the amount of L(t)> 0. Further, we assume that a quantity unit dL from the landscape L(t) has a value Vo. Production agent has a cost Ko associated with producing a service Two of value Vo.

Write a method in which a production agent that consumes resources from the dynamic landscape L(t) and producing a quantity value services Vo.

How can Production agent learn to be a stable production system?
How can production agent optimize the value of their production?

3.8 Active Particle Agents

Question: What are active particle agents?
Answer: Active particle agents are particulate agents with the ability to adapt to a state in the landscape It means that the agent assigned a free will to adapt to local conditions.

Question: What are examples of active particle agents?
Answer: Examples of active particle agents are particles are agents representing particles or objects with a buoyancy. It may be particle agents represented physical or organic materials, water or air. It may also be agents representing traffic, ships and the like.

Question: What can the particle agents in stationary landscape make of tasks?
Answer: Typical tasks are to identify complex flow patterns. Eg flow pattern in the water, waves etc.

LEARNING OBJECTIVES

Students should at the end of this lesson could:

1. Explain the applications of active particle agents
2. Develop a system model for an active particle agents
3. Planning an Experiment using active particle agents
4. Create a swarm active particle agents
5. Move particle agents affected by dynamic landscape

SYSTEMS THEORY

Meta system for a simple agent in a landscape can be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents an agent, L (t) landscape and N(t) is the relationships between agent and landscape. The network has a relationship N(t) = {A(t), L(t)} where A(t) represents an agent with stationary link between agent and landscape.

In this case represents L(x, y, z, t) a set of parallel landscape where each landscape may change in relation to time and place. constant. Landscape Model L(x, y, z, t) can for example contain a fixed terrain landscape L(terrain, x, y, z, t) and a power landscape L (power, x, y, z, t) where each position (x, y, z) has a current vector L.

Active particulate agent model A(t)

Active particulate agents have a purpose. It can for example be to follow a characteristic of the landscape. Agent purpose again determines particle agent properties. Agents can be modeled with the properties

$$A(t) = \{A(sheets, t), A(dyn, t), A(eti, t), L(learn, t)\}$$

Where A(sheets, t) represents the agent's architecture, A(dyn, t) the agent state dynamics, A(eti, t) agent ethics, and L(learn,t) agent learning strategy.

Agent architecture A(arc, t)

A typical agent architecture A(arc, t) for an active particle agent is:

$$A(arc, t) = \{Sensor, Body, Motor\}$$

Where sensor measures and updates the state of the landscape L(z, x, y, t), Body represents the agent form a visual agent. Engine is a service that maintains the agent's progress. Typical services for a particle agent will be:

A(Sensor, t) = {reader agent close environment}.
A(Body, t) = {Represents particle geometry and friction}
A Motor, t) = {Safeguards particle progress}

Agent dynamics A(Dyn, t)
Agent dynamics A(Dyn, t) represents the agent's state dynamics. A typical dynamic model for a particle model is:

Particle acceleration vector: a(Object,t+T) = A(Object, t) + F(Engine, t)/Mass
Particle force vector: F(Engine, t) = -K (Friction) v(Relative, t) + F(External, t)
Relative particle velocity: v(Relative, t) = v(Object) -v(Landscape, t)
Particle velocity vector v(Object, t + T) = v(Object, t) + T*A(Object, t)
Particle position vector: x(Object, t + T) = x(Object, t) + T*v(Obj, t)

Where T is the sampling interval. K(friction) represents frictional coefficient between the particle and particle relative velocity v(Rel).

Agent ethics A(Ethics, t)
Active particulate agents have a purpose. That conferred a free will to realize a goal that can be realized within a resource or another limitation or performance. It can be formulated as:

A(Ethics, t) = {Measure, Performance}

Where measure represents a desired state of the active agent particle. Performance represents the agent's resource or capacity to achieve the goal. By simulating active particle agents are performance related to the agent's architecture, state dynamics and learning ability.

Agent Learning A(Leather, t)
Agent learning is the realization of the agent's ethics where learning concept is linked to the characteristics

A(Learn, t) = {Identification, Control}

Where identification represents a measurement of the landscape of the agent's environment.

Control = (Target Identification) K

Control is thus how emphasize any deviation between target and identified condition

PROCEDURE

Agent Based Simulation of passive particle model can be considered as a laboratory experiment where the basis of the following procedure.

1. Set up a meta system model experiment
2. Set up an affiliate model
3. Formulate agent purpose and methods
4. Applications agent properties
5. Documents the agent properties
6. Discuss your results

Procedure for method development:

1. Formulated physical properties of the landscape L (t)
2. Formulate particle agent object and purpose of the landscape L (t)
3. Formulate a model for the agent engine and dynamic.
4. Formulate agent learning method for control of own condition.

EXERSIZE

Control Questions about particle agents.

1. Explain what an active particle agent means.
2. What can an active particle agent represent?

3. How can we create a physics model for an active particle agent?

Experiments with active particle agents
In this we will study how a group active particle agents, with a selected mass, adapts a desired height level in a closed landscape.

1. Assume the previous task and study how group active particle agents behave in the landscape.
2. Do some experiments where agents have random mass and adapt a random position.

3.7 Goal oriented Moveable Agents

Question: What are goal-oriented moveable agents?
Answer: It is agents who have the ability to move to specific targets in a landscape.

Question: What are the good examples of goal oriented moving agents?
Answer: Examples of goal oriented moving agents are agents representing people, cars, ships, etc. and have an itinerary between two or more points.

Question: What can goal oriented agents do?
Answer: In practical simulation of complex systems, is that agents should represent people, cars, ships, etc. The agents must be able to move from the established objectives in a more or less complex landscape. Targets in complex landscapes can be known, unknown, and it may be encumbered with obstacles. Some typical tasks for agents will be calculating travel times, costs, travel patterns and risk factors.

Question: How can agents reach targets in complex landscape?
Answer: The answer depends on the characteristics of the landscape. The easiest method is to set up a route through a set of objectives. So lets one agent follow the itinerary until the ultimate

goal. If the itinerary is not fixed via targets, the agent must follow the landscape and try out. If there are several targets to choose from, the agent must choose a target. This choice can be made on the basis of a value or cost consideration.

Question: How can an agent reach one of several targets in complex landscape?
Answer: The easiest method is to set values on target and then calculate the costs associated with achieving goals. The agent then selects the measure that provides the greatest value creation.

LEARNING OBJECTIVES

A discontinues this lesson

Knowledge

1. Could create a system model for goal-oriented agents
2. Could modeling own dynamics goal oriented agents
3. Could introduce control strategies for goal-oriented agents

Skills

1. Could utilize Unity game engine for simulation of goal-directed agents
2. Could program goal oriented
3. Be able to select control strategies for control of mole-focused agents

General knowledge

1. Have knowledge of applications to accomplish goals agents
2. Have knowledge about the possibilities and limitations of intentional agents

SYSTEM THEORY

Meta System

Meta system S(t) for moving agents can still be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents an agent, L (t) is the landscape and N(t) is the relationships between agents and landscapes. In this case, the agent A(t) a position (x1, y1, z1) in the landscape L (t) before it is moved to a position (x2,y2, z2).

Agent Model
Agent internal properties can be modeled with system perspectives

$$S(t) = \{A(Arc, t), A(Dyn, t), A(Eth, t), A(Lea, t)\}$$

where A(Arc, t) represents the agent's architecture, A(Dyn, t) agent dynamics, A(Eth, t) agent ethics and A(Lea, t) agent learning.

Agent architecture A(arc, t)
Moving agents can be modeled with the following services:

$$A(Arc, t) = \{Sensor, Body, Motor, Controller, Output\}$$

Where sensor measures the agent's position in the landscape, Body represents the agent's own dynamic characteristics Motor provides agent propulsion and Controller exercises control of the agent's speed and direction. Each service may have a set of specifications.

A(Sensor, t) = {Agent measurements to landscape or other agents}
A(Body, t) = {Visual model, volume, weight, friction ++}
A(Motor, t) = {Power, rudders, ++}
A(Controller, t) = {Control of speed, direction, start, stop, ++}
A(Output, t) = {Overrunning to landscape or other agents

Agent dynamics A(dyn, t)
Agents have their own dynamics that normally determines in accordance with balance models. Some typical dynamic models for a moving agent will be vectors:

Force: F(engine, t); Power Switch The dynamic model
Acceleration A(t) = F(t)/M (force / mass)
Friction: F(t) = - Kf * v(t) (force)
Speed: v(t +T) = v(t)+A(t) (m / sec)
Position: x (t + T) = x (t) + v (t) (z, x, y)

Agent ethics A(Eth, t)
Agent business ethics has characteristics A(Eth, t) = {Goals, Constraints}. The goal represents a target for the agent service and limitations represent characteristics of the agent's design or parameters.

Goal-oriented agents shall keep proper speed while they will reach the right target. They therefore have a more complex target structure than just moving agents. A typical target structure for a moving agent is:

1. State goal = (Speed, Direction, ++)
2. Position measuring = (Target position, Type, Value)
3. Behavior = (Position measures, State goal)

Position:
Agent position measures have target position defining an (x, y, z) position in a landscape. Type is a classification of what the target represents. Value is a classification of the target resource properties. Position goals collected in a vector.

State goal:
Here state goal are goals that are related goals for the agent's own dynamics. Typical tilstandsmål a measure of the agent's speed and direction. Physical dimensions are collected in a vector.

Behavior:
The term behavior is linked to changes in the agent's activities. Agent activities are in turn linked to the agent Target vector. Agent overall behavior is a set of position measures and state goal.

Agent Learning A(Learn, t)
Agent Learning A(Learn, t) is realizing agent ethics A(Eth, t).
Learning of the agent target structure is in this case:
1. Physical Learning: Learning of position and speed
2. Position Learning: Learning to reach the correct position
3. Conduct learning: Learning by order of objectives

The new here is thus introducing the concept of behavior, where a control function exercises control of the agent states.

Movement to one target:
A simple method to regulate the agent's position up to a target model is:

$$Vtx(t) = \log(Gtx^*|Etx(t)|)$$

Where Vtx(t) is the desired speed until manner Gtx is a reinforcement and |Etx(t)| is the agent's distance to the goal.

Target selection
If the agent must choose one among several goals, the agent must have a criterion for the selection of targets. A method for selecting targets is to select the targets that provide maximum value to the agent. This can make the rule

$$S(Position) = Max((Target\ position, Value) - Target\ Cost))$$

Where is Target Cost agent estimated cost to reach position.

Learning behavior
The term behavior is related to the agent's state between two or more targets in a target structure. Behavior control is to make sure the agent states in the target structure. This can be done in several ways. If the itinerary is known, we can create a target structure with itinerary, which controls a state machine, which in turn controls the agent states.

PROCEDURE

Case 1: One stationary target

System modeling

1. Set up a system model for agent architecture A(sheets, t)
2. Formulate agent ethics and set goals for the agent to reach a position
3. Set up the agent's target vector
4. Set up a strategy for the control of the agent's movement towards the target
5. Set up criteria for evaluating agent performance

Programming

1. Create the program Movable Agent
2. Introduce one Target vector in service A(Body)
3. Introduce control of the agent position in service in A(Controller)
4. Insert a cost function in the agent A(Body) that monitors the agent's deviation
5. Let A(Output) report the agent's speed, direction and costs

Discussion

1. Set up a test plan for how you will test and evaluate agent performance

EVENT 2: N-SUBGOALS

System modeling

1. Set up a system model for agent architecture A(Arc, t)
2. Formulate agent ethics and set goals to reach a position
3. Set up the agent's target vector with goals, type and values
4. Set up a strategy for the selection of targets with a maximum value
5. Set up criteria for evaluating agent performance

Programming

1. Create the program Movable Agent
2. Introduce one Target vector in service A(Body).
3. Introduce control of the agent position in service in A(Controller)
4. Insert a cost function in the agent A(Body) that monitors the agent deviations.
5. Let A(Output) report the agent's speed, direction and costs

Evaluation

1. Set up a test plan for how you will test and evaluate agent performance.

EVENT 3: MOVING TARGETS

System modeling

1. Set up a system model for agent architecture A(sheets, t)
2. Formulate agent ethics and set goals to reach a moving target
3. Set up the target vector with goals, type and values to the moving target
4. Set up criteria for evaluating agent performance

Programming

1. Assume the program Movable Agent
2. Introduce one Target vector in service A(Body)
3. Introduce a service A(sensor) that updates the position of the moving target
4. Let A(Controller) control agent speed and direction toward the moving target.
5. Let cost function monitors the agent's performance.
6. Let A(Output) report the agent's speed, direction and costs

EXERCISE

Control Questions about moving goal oriented agents

1. Explain what is meant by movable goal oriented Agents
2. What can movable goal oriented agents represent?

3. Explain how moving goal oriented agents can control its speed and direction toward a goal.
4. Explain how moving goal oriented agents can control its speed and direction toward a goal.
5. Explain how moving goal oriented agents can choose one of several targets.

Mobile goal oriented agents

In this task, we develop mobile goal oriented agent.

1. System Modeling: Create a system model S(t) for target-oriented agent.
2. Experiments with one goal: Create Movable Agent with parameter and methods for service so that it can move with constant velocity until a stationary target. Study agent performance by measuring ability to position itself to the target.
3. Experiment with a moving target: Create a new method that gives agents the ability to reach a movable must, which runs in a circle. Studying agent performance by measuring ability to position itself to the target.
4. Experiments with several objectives: Set up a set of goals, with random positions and values. Set up a rule for selecting targets with maximum value and study the agent's ability to optimize access to values.
5. Comment your results.

3.9 Intelligent agents

Question: What is an intelligent agent?
Answer: An intelligent agent is an agent with the ability to monitor and optimize performance.

Question: How can the agent monitor and optimize its own performance?
Answer: A common method is to compare the performance of cost functions. The principle here is that the agent selects the parameter

that provides the least cost from the cost function. This compares with a regulator, which adjusts a parameter to the least deviation.

Question: What are the typical applications of intelligent agents?
Answer: Intelligent agents are useful when the agent parameters are unknown. One can then hand over responsibility to the agents to customize their own parameters. Another example is when the agents optimize their own parameters to changes in environment. A typical example here is the control of speed of ships with changes in load or bottom conditions.

LEARNING OBJECTIVES

A discontinues this lesson

Knowledge

1. Be able to explain what an intelligent agent is something
2. Be able to explain what intelligent agents can be used to

Skills

1. Could utilize Unity game engine for simulation of moving agents
2. Could programming intelligent agents
3. Be able to select control strategies for intelligent agents

General knowledge

1. Have knowledge of applications to intelligent agents
2. Have knowledge about intelligent agents capabilities and limitations

TOPICS

1. System modeling of intelligent agents
2. System architecture for intelligent agents
3. Intelligent agents own dynamics
4. Cost Functions
5. Learning intelligent agents

SYSTEMS THEORY

Meta system S(t) for intelligent agents can still be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents an agent, L(t) is the landscape and N(t) is the relationships between agents and landscapes. In this case, the agent A(t) a position (x1, y1, z1) in the landscape L(t) before it is moved to a position (x2, y2, z2). Agent internal properties can be modeled with system perspectives

$$S(t) = \{A(Arc, t), A(Dyn, t), A(Eti, t), A(Lea, t)\}$$

where A(Arc, t) represents the agent's architecture, A(Dyn, t) agent dynamics, A(Eth, t) agent ethics and A(Lea, t) agent learning

Agent architecture A(arc, t)

Intelligent agents have in principle the same architecture as other moving agents with the properties

$$A(Arc, t) = \{Sensor, Body, Motor, Controller, Output\}$$

Where sensor measures the agent's position in the landscape, Body represents the agent's own dynamic characteristics Motor provides agent propulsion and Controller exercises control of the agent's speed and direction.

Agent dynamics A(Dyn, t)

Intelligent agents have in principle the same dynamics as other moving goal oriented agents. The difference is that they have a greater ability to optimize speed and direction from weight functions or costs.

Agent ethics A(Eth, t)

Agent business ethics has characteristics A(eti, t) = {goals, constraints}. That they will reach their own goals and simultaneously adjust own optimize their own limitations. It is then so that objectives and identification of constraints, are two sides of the same coin. When an intelligent shall reach a goal, you agent ethics formulated as

A(Eth, t) = {Variable goals, variable restriction}.

It becomes the optimization criterion that determines the agent's choice of objectives and constraints or properties.

Agent Learning A(Lea, t)

Agent Learning A(Learn, t) is realizing agent ethics A(Eth, t). Learning of intelligent agents is to optimize the agent performance or parameters. A typical model for the control of velocity is:

F(Engine, t) = (R(Speed, t) -v(Speed, t)) K(Speed, t) = e(t)K(Speed, t)

Where F(Engine, t) is the power control signals to the motor, R(Speed, t) is the desired speed v(Speed, t) is really speed and e(t) speed deviation. K(Speed, t) is the parameter for speed control that determines the agent's performance.

Monitoring performance

The agent performance can be calculated using a cost function. An example of a cost function is:

J(t) = e(t)Q + u(t)R

Where e(t) represents deviations from targets, Q is a chosen weighting or cost deviation, u (t) is the switch (power or energy) on services, R is selected emphases or cost of the switch. Is there uncertainty in the measurement of e(t) or u(t), it may be prudent to estimate the condition of the cost function J(t), to:

$$J(t+T) = La * J(t)+(1-La)[e(t)Q + u(t)R]$$

Where T is a sampling interval, La is a selected constant La <1, that determines how fast the J (t) to reach a mean value.

Customizing performance

Intelligent agents monitor its performance and then adjust its performance. In this case, the performance is related to speed and direction. There is an extensive literature on how to introduce a self-adjusting adaptation of the agent's performance. We shall here only indicate some simple methods:

Adaptive control

Some simple examples of adaptive control are:

$$K(Speed, t) = Ko * J (t) \text{ and}$$
$$K(Speed, t) = Ko * J (t) / (R (Speed) -J (t))$$

Der Ko is a selected constant and R(Speed) is desired velocity vector.

Control via neural network

Another method is to use a neural network as a self-regulating gain factor K(Speed, t). A neural network has characteristics

$$Nout = f(NN, Targt, Input)$$

Where NN represents the network's properties. Target are the condition to be identified, Input is the switch to the network and Nout is the response from the network. A control strategy, based on a neural network can then for example be implemented by setting

$$K(Speed, t) = f (NN, Targt = R(Speed), Input = v(Speed, t))$$

Network NN must then be trained continuously.

Control via Genetic Algorithm

Introduction of a self-tuning control can also be done with a genetic algorithm. Taking a PI controller can do this. A single control strategy S (urgency, t), based on a PI controller, can be formulated as.

$$K(Speed, t) = K0+1/x(t+1),\ x(t+1) = a*x(t)+e\ (t)$$

Der Ko represents the gain in the proportional, and a represents the integration factor in the integration clause. A genetic algorithm is Ko and a free variable as genes and optimized in relation to the cost function J (t) and deviation e(t).

PROCEDURE

System modeling

1. Set up a system model for agent architecture A(sheets, t)
2. Formulate agent ethics and set goals for the agent's speed and direction.
3. Set up the balance models for dynamics of mass, friction and disturbance.
4. Set up a strategy for the control of the agent's speed and direction
5. Set up a strategy for monitoring and evaluation of agent performance

Programming

Assume results from efforts goal oriented moving agents

EXERSICE

Control Questions about intelligent agents

1. What is an intelligent agent?
2. What can intelligent agents used?
3. How can intelligent agents modeled?

Experiments with intelligent agents

An intelligent agent should represent a moving object that moves in a landscape. The agent has a mass M and a friction coefficient Kf. A PID controller controls the desired speed of the movable agent.

1. Assume the previous task about goal oriented moving agents.
2. Set up a model for agent's properties with mass, friction and disturbance
3. Formulate an experiment in which the intelligent agent has the ability to update its own parametre.
4. Select a method for performance and monitoring the agent's performance.
5. Comment on the result of experiments

4 SOSIAL AGENTS

Question: What are social agents?
Answer: Social agents are a set agents who collaborate with the same rules within a common purpose.

Question: What are examples of social agents?
Answer: Examples of social agents are agents who represent a group of people, fish, cars, ships or particles that occur within a common set of rules.

Question: What can social agent do for you?
Answer: Social agents can solve optimization problem. They can for example simulate optimal production systems, logistics, identify optimum capacity in queues, traffic behavior models and individual-based market models.

Question: How can I model social agents?
Answer: Social agents can be modeled as a herd agents with a common set of rules. Every agent in the herd adapts when their activity compared to all the others, without any central control.

Demo videos
1. Swarm intelligence lecture: http://www.youtube.com/watch?v=IqhI4tWkyFc
2. Swarm lecture: http://www.youtube.com/watch?v=-G66iL__VdA

LEARNING OBJECTIVES

Knowledge

One should at the end of this lesson could:

1. Explain methods for system modeling of social agents
2. Explain how social agents can be used to solve puzzles

Skills

One should at the end of this lesson could:

1. Setting up an experiment using social agents that problem solvers
2. Programming groups with social agents
3. Introduce control strategies for social agents

General knowledge goals

One should at the end of this lesson:

1. Have knowledge about the application of social agents
2. Setting up an experiment with social agents

TOPICS

1. Agents groups and herds
2. Social agents architecture
3. Cohesion between social agents
4. Separation between social agents
5. Alignment social agents
6. Social agents in queues
7. Optimizing agent properties in queues

SYSTEMS THEORY

Social agents are part of the meta-system

$$S(t) = \{N(t)\ A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) a set of landscape and N(t) a set of relations between agents in a landscape. In this case, the relationship N(t) determined by the agent position relative to the other agents. Agent internal properties can be modeled with system perspectives.

$$S(t) = \{A(Arc, t), A(Dyn, t), A(Eth, t), A(Lea, t)\}$$

where A(Arc, t) represents the agent's architecture, A(Dyn, t) agent dynamics, A(Eth, t) agent ethics and A(Lea, t) agent learning.

Social agent ethics
A group agents A(t) are social agents when common social ethics

$$A(Eth, t) = \{Rules\}$$

Where Rules represent a joint of rules in agent group A(t).

Social agent architecture
Social agent architecture says something about how the agents are connected to each other via a common network. This can be formulated as

$$S(Arc, t) = \{N(t), A(t)\}$$

Where N(t) represented the network.

Social agent dynamics
Social agent dynamics is a concept that says something about how the whole system S(t) evolves over time.

Social agent learning
Social agent learning is methods to reach their own goals through a common set of rules.

4.1 Agent Swarm

Question: What is an agent swarm?
Answer: An agent swarm are social agents with a behavior that corresponds to the behavior of a swarm of birds fish or similar. A characteristic feature of agent swarms is that they have no other task than to keep the group together.

Question: What are examples of agent swarms?
Answer: Examples of agent swarms are simulation of groups of fish, people or particles, which have a total, flow movement in a landscape.

Question: What can the agent swarms do for you?
Answer: An agent swarm can show the overall behavior and structure from a group of agents. Normally it is allowing agents graze on a resource or fleeing from a cost. Moreover, it is so that a swarm is a basic model for studying other complex phenomena that queue until resources, market models, traffic models, risk models etc.

Question: How can I model agent swarms?
Answer: Modeling agent swarms are looking residue on three principles. One principle is that all agents in a group have common rules. The second principle is that all agents have access to the conditions of the other agents. The third principle is that there is no overall control. All agents operate independently of the others, but within the same rules.

LEARNING OBJECTIVES

Knowledge

One should at the end of this lesson could:

1. Explain methods for modeling of social agents
2. Explain how social agents can be used to solve puzzles

Skills

One should at the end of this lesson could:

1. Setting up an experiment using social agents that problem solvers
2. Programming groups with social agents
3. Introduce control strategies that collects social agents in a stable structure
4. Move social agents in a circle

General knowledge goals
One should at the end of this lesson:

1. Have knowledge about the application of social agents
2. Setting up an experiment with social agents

SYSTEMS THEORY

In this case we have a meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) landscape and N(t) is the relationships between agents in landscape. In this case, the relationship N(t) determined by the agent position relative to the other agents. Agent ethics A(eti, t) in this case is determined by the common set of rules agents adhere within the group A(t) with the agents.

Agent Model

Agent internal properties can be modeled with system perspectives

$$S(t) = \{A(arc, t), A(dyn, t), A(eti, t), A(lea, t)\}$$

where A(arc, t) represents the agent's architecture, A(dyn, t) agent dynamics, A(eti, t) agent ethics and A(lea, t) agent learning.

Agent architecture A(arc, t)

Moving agents can be modeled with the following services:

$$A(Arc, t) = \{Sensor, Body, Motor, Controller, Output\}$$

Where sensor measures the agent's position in the landscape, Body represents the agent's own dynamic characteristics Motor provides agent propulsion and Controller exercises control of the agent's speed and direction.

Agent dynamics A(Dyn, t)
Agents in a swarm normally goal oriented moving agents.

Agent social ethics A(Eth, t)
Social agents are based on the principle to adhere to a common set by rules. Every agent in the swarm reacts according to some simple rules, without any central control. The result of this is that the entire burst achieves a complex behavior. Agent ethics has an external and an internal perspective.

External ethics
Agent external ethics is attached to the group of common purpose. A common purpose may in turn be linked to a common regulatory framework.

A(External Eth) = {Cohesion, Separation, Alignment}

Where Cohesion represents a rule for coordinating agents, Separation represents a rule for separating agents and Alignment represents a rule for direction orientation of agents.

Internal ethics
Agent internal ethics are related to the agent's own goals. Some examples of the agent's own goals are:

A(Internal, Eth) = {Position, Costs, Criteria ,,,}

Agent social learning A(Learn, t)
Agents of social learning are adapting the agent state in relation to common social rules.

Cohesion
Learning Method for Cohesion is based on the agent first calculates the burst virtual center. This calculates each agent calculates its average value relative to all other agents. Cohesion center and then averaging this. This can be formulated as

$$Coh(Pos) = Total\ (A(j, Pos) - A(i, Pos)) / (Number-1)$$

Where A(i, Pos) is the agent in his position, which performs a calculation. A(j,Pos) are the positions of all other agents in the swarm. Each agent has a pull from his position and toward a common center. The distance to the virtual point, all the agents are

$$Coh(Dist) = Coh(Pos) - A(i, Pos)$$

Where Coh(Dist) is a distance vector. The direction is:

$$Coh(Dir) = ||Coh(Pos) - A(i,Pos)||$$

Where || || represents a direction vector of the distance. Cohesion force is a property of the agent. It says something about how much weight the agent put on having a distance from the virtual point. This power can be calculated with

$$F(coh) = Coh(dir) * Gcoh$$

where Gcoh represents the agent's control strategy

Separation

Learning Method for Cohesion is based on the agent calculates a counterforce, which is inversely related to the distance to the nearest agents. This can be done by different methods. A typical method is to calculate the first relative distance

$$relDist = A(j,Pos) - A(i,Pos)$$

and then divide by the square of the distance to

$$relDist = relDis/[(A(j, Pos) - A(i, Pos))\ (A(j, Pos) - A(i, Pos))]$$

The nearest agents will then give a maximum contribution to a relative distance relDist. Separation distance then becomes

Sum(relDist)

or the sum of the contributions from the nearest agents. That way, all the nearest agents contribute more or less with a separation force.

Alignment
Learning Method for Alignment is based on the agent calculates the mean velocity swarm swarms (velocity) for all change agents in the swarm. This can be formulated as

Swarms(Speed) = Total(A(j, Speed))/(Number-1)

where A(j, Speed) is the velocity of an agent A(j, t). Agent deviation, in relation to the burst speed is

Align(Speed) = (Swarm(Speed) -A(i, Speed))

Error deviation Alignment must then be corrected in the agent's progress

PROCEDURE

Procedure for method

1. Formulate first that experiment agents will perform for you
2. Formulate so how you want to test the results of the experiment
3. Formulate chosen methods.
4. If you use Cohesion, Separation, Alignment must think through what control strategy you will brukef

Procedure for programming
When calculating Cohesion, Separation and Alignment in Unity, each agent will have access to position and velocity of all other agents. It can be done through a real object, as in program list nobody is called controller. Via controller each agent access to the list of all the agents. Each agent then goes into the list, pulls out one by one

agent and swipes the agent state. See attached code sample UpdateSwarState (). Each agent can then make their own condition, in relation to the other, with their own methods

MANDATORY EXERCIZE

Control Questions about social agents

1. Explain briefly what is meant by the term Social agents
2. Explain briefly what is meant by the term agent swarms
3. Explain Craig Reynolds social rules for Cohesion, Separation and Alignment.

Experiments with social agents

It should be an experiment with social agents supplied a goal that goes in a circular orbit.

1. Create a movable goal oriented agent and let it move in a circle.
2. Create a swarm of agents in a landscape.
3. Create a method for agent Cohesion and study how agents are moving toward a common point. Adjust the force input to the agents and adjust to a natural speed.
4. Create a method for Separation and study how agents now stand in relation to each other. Adjust separation parameter to the agents get a suitable distance.
5. Create a method alignment and study how agent's direction now stands in relation to each other. Adjust alignment parameter to the agents get a fit control direction.
6. Create a common performance reference.
7. Comment on the result you have reached

4.2 Agent Queue

Question: What is an agent swarm queue?
Answer: An agent swarm queue is a set of social agents that compete for access to a resource.

Queue: What are examples of agent swarm queue?
Answer: Examples of agent swarms queues are simulation of traffic models where agents must pass an obstacle that restricts agents' speed in the flow direction. Other examples are agents that compete for access to a common goal.

Queue: What can simulate agent swarm queues do for you?
Answer: Simulation of swarm queues can be used to solve optimization problem. It can be to identify the capacity for a barrier forming a queue, identifying the expected number in a queue, or it may be to identify strategies to pass through a line.

Q: How can I model agent based swarms queues?
Answer: Silence queues can be modeled as social agents. They have common rules with Cohesion, Separation and Alignment that unifies queue. In addition seeks every agent a goal. This goal may be common to all agents.

LEARNING OBJECTIVES

Knowledge

One should at the end of this lesson could:

1. Explain methods for modeling of queues with social agents
2. Explain how social agents that queues can solve puzzles

Skills

One should at the end of this lesson could:

1. Setting up an experiment using social agents in queues
2. Program a queue of social agents
3. Introduce strategies for learning agents to adapt see a queue situation
4. Introduce strategies for optimizing agents in a queue.

General knowledge goals

One should at the end of this lesson:

1. Have knowledge about the application of social agents
2. Setting up an experiment with social agents

SYSTEMS THEORY

Meta system S(t)

In this case we have a meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) landscape and N(t) is the relationships between agents in landscape. In this case, the relationship N(t) determined by the agent position relative to the other agents.

Agent ethics.

Agent business ethics in a queue can be formulated as:

$$A(eti, t) = \{Cohesion, Separation, Alignment, Size\}$$

We see here that agents in a queue must adhere to the social rules Cohesion, Separation and Alignment. In addition, there now entered the agent's Goals as a new rule. Targets can then be several things. It may be desired position, a desired value or any agent would like to optimize.

Agent Learning

Agents in a queue must here learn to exercise four types of learning, at a time.

1. Learning of the agent toward this goal
2. Social learning of Cohesion, Separation, Alignment
3. Methods of learning tend to follow classical control theory.

Proportional regulator

The method here is based on Control Act

Force = (Target Position) K

Where Forces represent a motor actuation, Measure is an a position measures, Position is the agent position and K is a selected constant

Adaptive regulator
The selected constant K can be replaced an adaptive regulator. If one chooses to use a neural network to implement an adaptive controller, a set:

Force = NN(Target Position)

Where NN() represents a neural network, which in the simplest case can be a neuron. If one chooses to use a genetic algorithm to implement an adaptive controller, a set:

Force = (Target Position) K (t)

Where K(t) represents a gene. The agent can then create a population with K(t) genes. It chooses so in real time the gene that produces the least square deviation.

PROCEDURE

Modalities for the method:

1. Formulate first that experiment agents will perform
2. Formulate so how you want to test the results of the experiment
3. Formulate methods so the agent will use to achieve goals.
4. Formulate emphasis on the social characteristics of Cohesion, Separation and Alignment

Method of programming
Here you combine the procedure you used with Movable agents and procedure with Social agents.

EXERCISE

Control Questions about social agents in queue

1. Explain methods for modeling of queues with social agents
2. Explain how social agents that queues can solve puzzles

Experiments with social agents in queue
It should be an experiment with social agents in a queue.

1. Create a landscape where agents must pass through a hole in a wall
2. Let the goal agent move through the hole in the wall, so that it is drawing the other
3. Study how this experiment influences a good balance of parameters for control of Cohesion, Separation and Alignment.
4. Comment on the result
5. How can a train here the agent's parameters with a Genetic Algorithm?
6. How can a train here the agent's parameters with a neural network?

4.3 Swarm Nest building

Question: How do Swarm create new swarms?
Answer: Swarm agents are grazing on resources in one landscape as long as there are resources. The they moves over to new resources in new landscapes.

Question: What are examples of swarms to new resources?
Answer: Some typical are swarms representing market models, until grown by cities, the spread of populations and changes in ecosystems.

Question: What can new entrants swarms do for you?
Answer. They may be used to simulate the dispersion stream under different conditions.

Question: How can we simulate new entrants with swarms?
Answer: A typical example is ant algorithm. Some agents take a random trip for new resources. The most fortunate leaves most tracks. Then the rest will follow.

Biological agents
1. Fish swarm: http://www.youtube.com/watch?v=UM8SzF6_0sM&feature=related
2. Ant agents: http://www.youtube.com/watch?v=C1XYMG5R5cQ&feature=related

SYSTEMS THEORY

Meta system S(t)
We have a meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L (t) landscape and N(t) is the relationships between agents in landscape. In this case, so that a set of agents A(t) is feed on a set of resources in the landscape L (t) via a network N(t). Establishment swarms are based on swarm A(t) seeks a new set of resources ii countryside L (t).

Particle agent ethics
A social agent A(t) that makes an market entry has several social rules. Typical rules

$$A(eti, t) = \{Social\ rules, search\ Rules\ Following\ Rules\}$$

Where Social rules are rules related group together. Typical rules Cohesion and Separation depending on the application.

Simulation of passive flow models:

By simulation of flow models have a happy:

1. Social policy oriented as separation
2. Following Rules relating to the powers of external landscape

Simulation of active particles in the flow models:
Active particles flow models may, for example represent objects in water. Typical models here is:

1. Social policy oriented separation
2. Goal-oriented rules for customizing environment Following Rules relating to the powers of external landscape

Simulation of optimization problem:
By simulation optimization problem has a like:

1. Social policy oriented as cohesion towards optimal ranges
2. Social policy oriented as separation from local agents

Social agent learning
Social agent learning are methods agents use to realize the burst ethics. An important principle here is that there is no central control. All learning and control are linked to agents' individual models.

Learning social rules
Learning of Social rules are based on estimating of parameters and control of power control signals between moving agents

4.2 Particle swarms

Question: What do particle swarms mean?
Answer: A particle agent is an agent who represents a passive object. A group particle objects behave as a particle swarm when they moved in a landscape.

Question: What are examples of particle swarms?
Answer: Particle swarms represent like physical objects. Some examples are floating objects in water, oil, water drops and the like.

In agent-based modeling, particles can also represent goods, networked computers, cars, ships and the like.

Question: What can the particle swarms do for you?
Answer: A typical task for particle swarms is to identify flow patterns. Another task is to solve optimization problem.

Question: How can a simulate particle swarms?
Answer: A typical example is ant algorithm. Some agents take a random trip for new resources. The most fortunate leaves most tracks. Then the rest will follow.

LEARNING OBJECTIVES

Knowledge

One should at the end of this lesson could:
1. Explain "Ants Algorithm" as the method
2. Explain what "Ants Algorithm" can be used to

SYSTEM THEORY

Meta system S(t)
We have a meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) landscape and N(t) is the relationships between agents in landscape. In this case, so that a set of agents A(t) is feed on a set of resources in the landscape L(t) via a network N(t). Establishment of swarms is based on swarm A(t) that seeks a new set of resources ii countryside L(t).

Social Agents Ethics

Social agents A(t) have several social rules. Typical rules

A(eti, t) = {Social rules, search Rules Following Rules}

Where Social rules are rules related group together. Typical rules Cohesion and Separation. Search Rules are rules as single agents use to search for new resources. Tracing Rules are rules the agents use to monitor track between different resources.

Social Agent Learning
Social agent learning are methods agents use to realize the burst ethics. An important principle here is that there is no central control. All learning and control are linked to agents' individual models.

EXERCIZE

1. Explain the "Ants Algorithm" as the method
2. Explain how "Ants Algorithm" may be used to solve optimalization problems

4.3 Particle swarm optimization

Question: What is meant by particle swarm optimization?
Answer: There are a swarm of particles that together solve an optimization problem. Particle swarm optimization, is also associated with the algorithm Particle Swarm Optimization (PSO), an algorithm that is widely used to solve optimization problem.

Question: What are examples of PSO?
Answer: PSO is used to optimize control, communications, logistics, control of real-time systems, energy management, etc. It has thus show itself to be a smart way to solve a wide variety of types of tasks.

Question: What can the particle swarms optimize for you?
Answer: A typical task for particle swarms is to identify flow patterns that are limited by local conditions. In this lesson we will look at how

a PSO can be used to simulate traffic through a complicated road system.

Question: How can a make a particle swarm optimization?
Answer: It is really very simple. PSO is essentially a variant of the algorithm for social agents. Cohesion is replaced by a global goal, alignment is taken away, and separation is connected to a local charge.

LEARNING OBJECTIVES

Knowledge

One should at the end of this lesson could:

1. Explain "Particle Swarm Optimization (PSO)" as the method
2. Explain what "PSO" can be used to

Skills

One should at the end of this lesson could:

1. Do an experiment with a PSO algorithm
2. Evaluate own results

SYSTEMS THEORY

Meta system S(t)
We have a meta system

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, (t) landscape and N(t) is the relationships between agents in landscape. In this case, so that a set of agents A(t) is feed on a set of resources in the landscape L(t) via a network N(t). Establishment swarms are based on swarm A(t) seeks a new set of resources ii countryside L(t).

Agent ethics

The social agents A(t) have several social rules. Typical rules

A(Eth, t) = {Search} Rules

A particle agent has no social rules. That agent has no common rules with other agents in the swarm. PSO swarms have in principle only two search rules. The one rule seeks global goals, the second rule seeks local targets. It's so random weighting of rules that determine what has the greatest impact.

Agent learning
PSO normally no learning in real time. The learning is normally based on one uses is the population with random parameter. Then see a parameter, which turns out to be good.

Learning with Particle Swarm Optimization (PSO)
PSO algorithm is based on a simplification of Craig Reinhold's social rules for Cohesion, Separation and Alignment. The algorithm is based on the rule:

Vij(t + 1) = C1*Vij(t) + C2*R1*[L(Best local, t) - Lij(t)] + C3*R2*[L(Best global,t) - Lij(t)]

Xij(t + 1) = Xij(t) + Vij(t + 1)

Where Xij(t) is the agent(i, j) position at time t, Lij(t) is a landscape position, Vij(t) is the agent's rate of landscape L(t), [C1, C2, C3] are parameters chosen and [R1, R2] are random parameters between [0, 1]. L(best local, t) is the best local position and P(Best global, t) represents the best global position

MANDATORY EXERCISE

Knowledge Questions

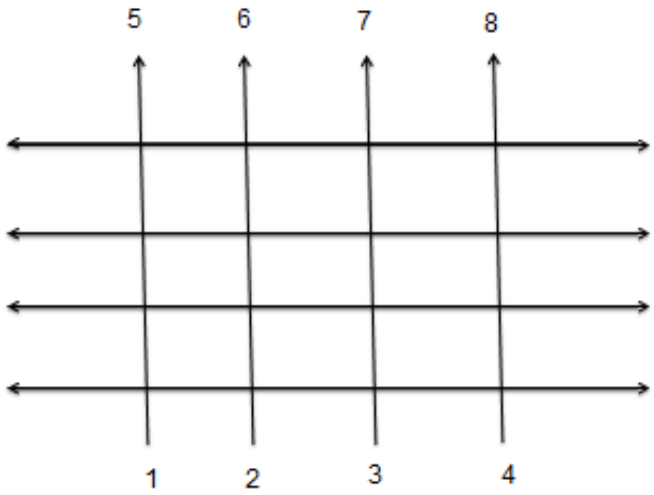
1. Explain how to model particle swarms
2. Explain what particle swarm can be used

3. Explain the principles of Particle Swarm Optimization algorithm

Cooperative Intelligent Traffic System

Vegsystem med gatekryss

HØGSKOLEN I ÅLESUND



In this exercise, will use the PSO algorithm to simulate a traffic pattern in a landscape. We have a landscape with a road system as shown, with input 1-4 and output 5-8. In connection with this landscape, we want to study what will happen when particulate agents have a random input 1-4 and a random starting 5-8. In this simulation particle agents have their global objectives and local costs associated with the position in the landscape.

1. Formulate the experiment as a lab experiments
2. Create at System model for the experiment
3. Set Agent rules for the experiment
4. Set rules for performance.
5. Set PSO rules for the experiment

6. Formulate test criteria for the experiment
7. Program a traffic system landscape
8. Program and run agents.
9. Verify the result
10. Explain the results

Literature

http://en.wikipedia.org/wiki/Particle_swarm_optimization
http://cswww.essex.ac.uk/technical-reports/2007/tr-csm469.pdf

4.4 ProCumer Agents

Question: What are stationary agents?
Answer: Stationary agents are agents with fixed positions in a landscape.
Question: What are examples of stationary agents?
Answer: Some examples of stationary agents are agents representing plants, trees, buildings, industrial production systems etc.

Question: What can stationary agents do for you?
Answer: Stationary agents can solve optimization problem. Typical examples are to determine optimal resource utilization, production capacity, timing for optimal growth, optimal market etc.

Question: How can model stationary agents?
Answer: Stationary agents are normally consumers and producers of a resource in a landscape. Agent optimization problem is then to adapt their cost and production capacity in relation to local resources

ProCumer Agents in a value chain

Question: What are stationary ProCumer agents in the value chain?

Answer: Stationary agents in a value chain are agents, which further refines resources from other agents.

Question: What are examples of agents in value chains?
Answer: Examples of agents in value chains are industrial food chains, economic systems and ecological systems. In industrial production is it that anyone who produces something is linked to a food chain. That everyone has one or subcontractor of goods, capital and services, and all has one or another customer who receives the goods or services produced. This range of services can be regarded as a series production of value-added services.

Question: What can agents in value chains do for you?
Answer: Agents in value chains can solve optimization problem. Typical examples are the optimization of production capacity, optimum time of production and optimal pricing.

Question: How do we model a supply chain of ProCumer agents?
Answer: In a value chain is all agents both consumers and producers. Meanwhile, production associated with a cost. This cost must be added as a value in the production chain. That means that all agents must optimize its cost in relation to the production value on consumer side and on the production side.

Systems theory

Meta system in this case may be expressed as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L (t) landscape and N(t) is the relationships between agents and landscapes.

Network N(t)
The network has a relationship N(t) = {A(t), L(t)}. In a series of production is A(t) = {A(1, t), A(2, t),,,, A(N, t)} a set of agents which

each agent has a stationary link between agent and landscapes. The network of communication between agents about can be linked directly between agents and through landscape. This can be done most easily when an agent A(1, t) consume from landscape cell L(1, x1, y1) and produces the landscape cell L(2, x1, y1). Agent A(2, t) consume from cell L(2, x1, y1) and produces the cell L(2, x2, y2), etc.

Landscape L (t)
We assume here that the landscape L(t) is represented with a 3D matrix with a set of landscape L(n, x, y) where L(1, x, y) has a growth model as described previously.

Agent Model A(t)
In this case represents the A(t) a set of agents

A(t) = {A(1, t), A(2, t),,,, A(N, t)}

where each agent has the same architecture as previously. In addition, there is an identification number for each agent.

Agent services
A(Sensor, t) = {Collect resources from landscapes. Setting production landscape}

A(Body, t) = {A simple geometry}
A(Produce, t) = {A linear transformation y = a * x with linear cost}
A(Check, t) = {Control of consumer}
Agent ethics A(eti, t)

Every agent in the population has the same ethics:
A(Eth, t) = {To optimize their own consumption in landscape over time}

Agent ethics A(Learn, t)

Every agent in the population here has also the same method for the control of the agent consumption.

The Game
The game in this meta system is that each agent in the production chain must adapt his or her own consumption for consumption and production in the preceding agent in the production chain. This is expected to affect overall system dynamics.

Task
Select the appropriate values of the reference R and K control, and maintain the set of agents that exert a serial production of a landscape.

Question
How is the stability of consumption and production in the value chain?
How can one find the optimal values of R and K?

ProCumer Agents in a marked

Question: What are ProCumer agents in a market?
Answer: ProCumer agents in a market are agents competing for a resource.

Question: What are examples of ProCumer agents in a market?
Answer: All ecological, industrial and social activities associated with one or another form of parallel competition between activities. In series based production we saw that the agents could adapt by adjusting production capacity after values. In a parallel production is not possible. The agent must then adapt its costs by adapting its capacity or framework.

Question: What can ProCumer agent in a market do for you?
Answer: They can find a balance between cost, capacity, price in relation to external competitors etc.

Question: How can I make a model of stationary agents in a market?
Answer: One can let the agents take decisions based on game theory

Systems Theory

Meta system S(t) in this case can also be formulated as:

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) landscape and N(t) is the relationships between agents and landscapes.

Landscape L(t)

Landscape with pricing model is the same as previously.
Furthermore, the agents located in each their landscape position L(x, y).

Network N(t)

Agent network N(t) = {A(t), L (t)} is in this case so that all the agents in the population A(t) is connected to the same cell L(x, y) in landscape L(t).

Agent Model

Agents can be modeled as previously properties

$$A(t) = \{A(Arc, t), A(Dyn, t), A(Eth, t), L(Learn, t)\}$$

Where A(sheets, t) represents the agent's architecture, A(dyn, t) Agent state dynamics, A(Eth, t) agent ethics, and L(Learn, t) agent learning strategy.

Agent services:

A(Sensor, t) = {Collect resources from landscapes. Setting production landscape}. A(Body, t) = {A Simple geometry}.

A(Producer, t) = {A linear transformation y = a*x with linear cost}.
A(Controller, t) = {Control of manufacturing value}

Agent ethics A(Eth, t)
Every agent in the population has the same ethics.
A(Eth, t) = {To produce when it has an increased value}

Agent Learning A(Learn, t)
In series based production task was to adapt a production volume of a production value in accordance with the cost function

$$J(t) = Q*X(t) + P*L(t)$$

where X(t) represents a state vector in agent A(t), Q represents the emphasis or the cost of the condition, L(t) the agent network to the landscape L(x, y, t) and P resource access or costs associated landscape.

When multiple agents have parallel access to the same resources and values, only those agents that have adapted production costs, could exploit their production. The agent must then learn what is properly adjusted cost matrix Q and P relative to the value set in the countryside.

Learning with Genetic Algorithm
One method to determine the appropriate costs are learning cost function with a genetic algorithm. A typical procedure is then to:

1. Let the cost elements q1 and q2 be free genes.
2. Generate a population of agents with random values on genes.
3. Remove the agents with the best cumulative object function.
4. Cross the genes from the best agents.
5. Generate a new agent population with new genes.

The Game
The game in this meta system is that all ProCumer agents must adapt its production capacity and costs by price and capacity in the

market. In this game, it is expected that there will be a more complex dynamics influenced by the number of agents who survive in the market.

EXERCISE

Create a set of agents in a parallel production where each agent regulates its own production capacity in accordance with a pricing model and a cost model.

Question

1. How is the stability of prices and production?
2. How is the stability of the agents' cost functions?

5 EVOLUTIONARY AGENTS

Question: What are evolutionary agents?
Answer: Evolutionary agents are agents, which are able to adapt to better performance in a changing landscape.

Question: What are examples evolutionary agents?
Answer: Evolutionary agents may be useful in modeling all kind of biological and technological applications.

Question: How are the agents used in an application?
Answer: There are two kinds of application of evolutionary agents. The first is to start with a population of random agent, and the let the agents adapt to a stationary landscape. The second method is to let the agents adapt to a time variant landscape.

Question: What is principle behind evolution agents?
Answer: The basic principle is to let each agent learn from the best agents and then add a random property.

Learning Objectives

Knowledge

One should at the end of this lesson could:

1. Explain methods for modeling of evolutionary
2. Explain a system model for evolutionary agents

Skills

One should at the end of this lesson could:

1. Setting up an experiment using evolutionary that problem solvers
2. Programming groups with evolutionary agents
3. Introduce a fitness strategy for evolutionary agents

General knowledge goals
One should at the end of this lesson:

3. Have knowledge about the application of evolutionary agents
4. Setting up an experiment with evolutionary agents

Systems Theory

In this case we have a metasystem

$$S(t) = \{N(t), A(t), L(t)\}$$

where A(t) represents a set of agents, L(t) landscape and N(t) is the relationships between agents in landscape. In this case, the relationship N(t) determined by the agent position relative to the other agents. Agent ethics A(eti, t) in this case is determined by the common set of rules agents adhere within the group A(t) with the agents.

Agent Model

Agent internal properties can be modeled with system perspectives

$$S(t) = \{A(Arc, t), A(Dyn, t), A(Eth, t), A(Lea, t)\}$$

where A(Arc, t) represents the agent's architecture, A(Dyn, t) agent dynamics, A(Eth, t) agent ethics and A(Lea, t) agent learning.

Agent architecture A(Arc, t)

Moving agents can be modeled with the following services:

$$A(Arc, t) = \{Sensor, Body, Motor, Controller, Output\}$$

Where sensor measures the agent's position in the landscape, Body represents the agent's own dynamic characteristics Motor provides agent propulsion and Controller exercises control of the agent's speed and direction.

Evolutionary Agent dynamics A(Dyn, t)

Evolutionary Agents has a dynamic state related to variable parameter.

Evolutionary Agents ethics A(Eth, t)
Evolutionary agents ethics are related to a fitness function or a cost function.

The GA method

Evolutionary agent may be based on a Genetic Algorithm, based on the following method.

1 Set the quality criteria.
Quality criteria may be a fitness function, a cost function or an error function.

2. Select a gene
A Gene vector is the set of free variable parameter that has to be tuned to an optimum state. Typical gene variable is parameter that controls speed, direction, target value and more.

3. Introduce an optimization procedure
A typical optimization procedure is:

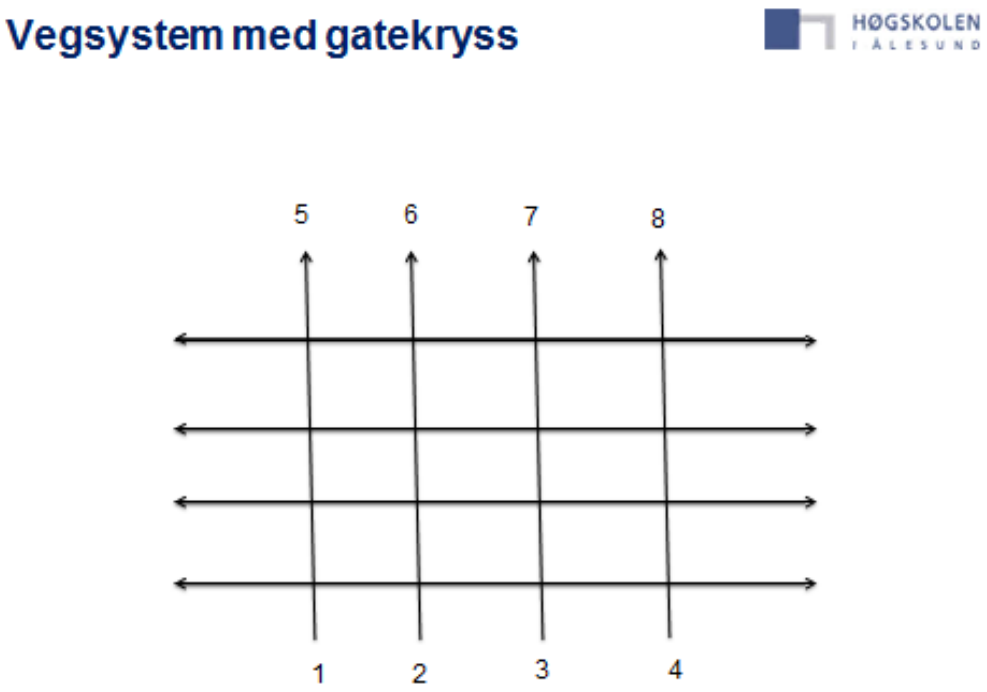
1. Create a population of agents that have random genes
2. Run the agent population
3. Select 20% of the best agents
4. Produce a new population from the best, by crossing genes
5. Introduce a random mutation on 5% of the genes
6. Go to 2
7. End when the population have a satisfied performance

Mandatory Exercise

Knowledge Questions

4. Explain how the GA algorithm may be used to adapt agents to a time variant landscape
5. Explain the GA algorithm
6. Explain typical applications of Evolutionary agents

Evolutionary agents in Cooperative Intelligent Traffic System



In this exercise, we shal use the PSO algorithm to simulate an evolution traffic in a landscape. We imagine that we have a landscape with a road system as shown, with input 1-4 and output 5-8. In connection with this landscape, we want to study what will happen when particulate agents have a random input 1-4 and a random starting 5-8. In this simulation the particle car agents have their global objectives and local costs associated with the position in

the landscape. In this exercize the PSO parameter is based on a GA.

1. Formulate the experiment as a lab experiments
2. Create at System model for the experiment
3. Set Agent rules for the experiment
4. Set PSO rules for the experiment
5. Introduce genes for PSO parameter
6. Introduce a cost for time consumes, agent collision, and agent outside the road track.
7. Program and run agents.
8. Verify the result.
9. Explain the results.

6 PROGRAM SCRIPT EXAMPLES

Hallo Agent

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class Agent : MonoBehaviour {
// Global data variable
public int myNr;               // Agent Number
public string myName="Bond";// Agent Name
public float waitTime=1.0f;      // Agent wait time
public AgentServer controller; // Connection to AgentServer

      // Start process
      void Start() {
            // Agent start methods
            StartCoroutine(UpdateState());
      }      // End Start

      // Runtime process
      IEnumerator UpdateState() {
       while(true) {
       // Agent runtime methods
      ReadMe();
      yield return new WaitForSeconds(waitTime);
         } // End while
      } // End Update States

      void ReadMe() {
      Debug.Log("Hello, my name is "+myName+" Nr "+myNr);
      }// End Start
}
```

Hallo Agent Server

Kode for AgentServer(script)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
public class AgentServer : MonoBehaviour {
// Global variable
public Agent aPrefab;     // Agent prefab connection
// A list of Agents
public List<Agent> Agents=new List<Agent>();
public int agentNr;                  // Agent number
public int maxAgents=10;               // Max agents in collection
public float waitTime=2.0f; // Wait time in seconds
      void Start () {
            // Stat methods
            CreateAgents();      // Create a set of agents
            CreateAgents();
            StartCoroutine(UpdateState());
      }

      IEnumerator UpdateState() {
      while(true) {
      // Server methods
      yield return new WaitForSeconds(waitTime);
      } // End while
      }// End Update States

      void CreateAgents () {
            for (int i = 0; i < 10; i++){
            // Creates an agent at current position and rotation
            Agent anAgent = new Agent ();
            anAgent = Instantiate(aPrefab, transform.position, Quaternion.identity) as Agent;
            anAgent.myNr=i;
            // Set agents to a random position inside the swarm sphere
            anAgent.transform.parent = transform;          // Set agent parent position
            anAgent.transform.localPosition = new Vector3(    // Set agent local position
            Random.value * collider.bounds.size.x,          // Set random bound x-siomze
            Random.value * collider.bounds.size.y,          // Set random bound y-size
```

```
            Random.value * collider.bounds.size.z) - collider.bounds.extents;
            Agents.Add(anAgent);                    // Set agent to agent list PUT
            } // End for
        } // End Create Agents

        void ReadMe() {
            Debug.Log("Read nr: ");
        }       // End Start
} // End program
```

Update: Swarm State

```
void UpdateSwarmState() {
// --- Reset Variable ----
        cohCen=Vector3.zero;
        Vgro = Vector3.zero;
        sepDis = Vector3.zero;
        //tarDis= Vector3.zero;
        N=0;
        foreach (GeneralAgent anAgent in controller.myAgents) {
            if (anAgent == this) {       // To all agents exept to this agent

            }// End if
            else {
// === COHESION ================================
            // --- Sum of Agent positions
            cohCen = cohCen+anAgent.transform.localPosition;

// === SEPARATION ============================
            // --- Sum of Agent distence
            // --- Distance to next agent
            //relDis= myPosition-anAgent.transform.localPosition;
            relDis= X-anAgent.transform.localPosition;

            // --- Distance adjusted
            relDis = relDis/(relDis.sqrMagnitude+0.1f);
            // --- Sum of relative distances
            sepDis = sepDis+relDis;

// === ALIGNMENT =================
            // --- Sum of Agent speed
            Vgro=Vgro+anAgent.rigidbody.velocity; // Speed sum
            N=N+1;    // Agent numbers in swarm
            }       // End Eles
        }       // End Foreach Agent
```

```
// === COHESION ======
        cohCen=cohCen/(N-1);   // Mean Cohesion Position
        cohDis=(cohCen-X);       // Distance to Cohesion Centre
        cohDir=cohDis.normalized;// Direction to Cohesion Centre

// === SEPARATION ====
        //sepDis=sepDis/(N);

// === ALIGNMENT ====
        Vgro=Vgro/(N-1);                // Mean Agent Speed
        aliDir=Vgro.normalized;         // Mean Adent Speed Direction
}// End UpdateMyFlockState
```

Time Server

```
using UnityEngine;
using System.Collections;
public class TimeServer : MonoBehaviour {
        // Time values
    public int i;
    public float simTime, playTime, aTimeScale, curTime, offsetTime;
    public float StartTime;
    public float [] RealTime, SimTime,  EndTime;
    public string [] TimeTxt; // (sec, min, hr, day, yr)
    public float dTime=1.0f;
        private Rect _win1Rect;

        // Window values
        private float maxTimeSpeed=100.0f;
        private float minTimeSpeed=0.0f;
        private float scrollPos=1f;

        void Start() {
                setStartTime();                 // Sett timme init
                setGUI();                       // Set and Start GUI
                StartCoroutine(startClock());               // Start clock process

        } // Set all start conditions

        void Update() {
                //countFrames();
```

```
        //countTime();

}  // Computes for each frame



void setStartTime() {
        TimeTxt = new string [] {"sec","min", "hr", "day", "yr"} ;
        RealTime =new float [] {0,0,0,0,0} ;
        SimTime =new float [] {0,0,0,0,0} ;
        aTimeScale=1f;
        playTime=0f;
        StartTime=Time.time;
}       // set start time condition

void setGUI() {
        // Set GUI menues (x1,y1,x2,y2)
        _win1Rect = new Rect(20,20,120,160); // Set GUI window
}       // set GUI for time control

IEnumerator startClock() {
    while(true) {
        // Update my Clock state
        playTime=playTime+dTime;    // Uptate playtime
        countTime();                // Update simtime
        yield return new WaitForSeconds(dTime);
    } // End while
}       // Time control window

void countTime() {
        curTime=Time.realtimeSinceStartup;
        simTime=aTimeScale*playTime;     // Simtime in seconds
        SimTime[3]=(int)(simTime/(24f*60f*60f) % 24f);     // Days
        SimTime[2]=(int)(aTimeScale*playTime/(60f*60f) % 60f);          // Hours
        SimTime[1]=(int)(aTimeScale*playTime/60f % 60f);                // Minutes
        SimTime[0]=(int)(aTimeScale*playTime % 60f);                    // Seconds
}       // End countTime

void OnGUI() {
```

```
        // Draw a window
        string guiHeadLine="Time Scaling";
        _win1Rect=GUILayout.Window(1,_win1Rect,DrawWin1,guiHeadLine);
    }   // End OnGUI

    void DrawWin1(int winId) {

        float oldScrollPos=scrollPos;
        // Get window scroll position
        scrollPos=GUILayout.HorizontalScrollbar(scrollPos,1,minTimeSpeed,maxTimeSpeed);
        aTimeScale=scrollPos;
        // Set information to the window
        GUILayout.Label("Play Time: "+playTime);
        GUILayout.Label("Time Scale: "+aTimeScale);
        GUILayout.Label("Sim Time: ");
        GUILayout.Label("  "+"day, hr, min, sec");
        GUILayout.Label("  "+SimTime[3]+"      "+SimTime[2]+"     "+SimTime[1]+"    "+SimTime[0]);
        GUI.DragWindow(); // Display window information
        if(scrollPos != oldScrollPos) {
            // Scaling the internal simulation timer
            Time.timeScale=aTimeScale;
        }  // End Time scaling
    }  // End DrawWintranslation

} // End program
```

Whacher

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
//    === Whacher  ============
//    Processes
//    Input
//        Target
//    Output
```

```
//            my ttransform
//  Project: Virtual More
//     Autors:
//     Original authors: Unify Community and
//  Benoit Benoit FOULETIER,
//  Modified by: R. Bye
//     Modified by H.Yndestad
//  Last update 09.10.2009
//     By H. Yndestad
//     ============================
public class Watcher : MonoBehaviour {
      public AgentSystem aTarget;
      //public Target aTarget;
      public Vector3 myTarget,flockCenter;
      void LateUpdate() {
            if (aTarget) {
                  // my posision = flockcentre pos + target pos
                  //myTarget= aTarget.flockCenter+aTarget.transform.position;
                  myTarget=GameObject.Find("Target").transform.position;
                  flockCenter=aTarget.flockCenter;
                  //myTarget= (aTarget.flockCenter+aTarget.transform.position);
                  //myTarget= (aTarget.flockCenter);
                  myTarget= aTarget.cameraPos;
                  transform.LookAt(myTarget); // Update my position
                  //transform.position(myTarget); // Update my position
                  //print("Camera Target = "+ myTarget);
            }     // End if
      }     // End LateUpdate
}  // End Watcher
```

Move In Circle

```
using UnityEngine;
using System.Collections;

public class MoveInCircle : MonoBehaviour {

      // Move target around circle with tangential speed
```

```
    public float tangentialSpeed;   // m/s
    public float circumference;     // m
    public float targetRadius;          // m
    public float period;            // s
    public float angularSpeed;          // rad/s
    public float currentAngle;          // rad/s
        Vector3 Centre=new Vector3();

    // Use this for initialization
    void Start () {
        tangentialSpeed = 6f;
        circumference = 600f;
        targetRadius = circumference/(2*Mathf.PI);
        period = circumference/tangentialSpeed;
        angularSpeed = 2*Mathf.PI/period;
        currentAngle = 0f;
        Centre=new Vector3(1000,280,1000);
    }


    // Update is called once per frame
    void Update () {

        transform.localPosition = Centre+ targetRadius*(
            new Vector3(Mathf.Sin(currentAngle), 0, Mathf.Cos(currentAngle)));

//          rigidbody.velocity = new Vector3(Mathf.Cos(currentDeg), 0, -Mathf.Sin(currentDeg));

        currentAngle += angularSpeed*Time.deltaTime;

        if (currentAngle > 2*Mathf.PI) {
            currentAngle = currentAngle-2*Mathf.PI;
        }
    }
}
```

Land Server

```
using UnityEngine;
using System.Collections;
public class LandSystem : MonoBehaviour {
// --- External Network ---
public TimeServer aTimeServer;
public float dTime;
//public ClimateServer aClimateServer;
//public SunServer aSunServer;

//public SeaServer aSeaServer;

// --- Abstract Landscapes ---
// --- Physics data ----------
public float [] SunRad;          // Sun Radiation
public float [] AirTemp;    // Air temperature
public float [] AirRain;
public float [] AirWind;    //
public float [] AirO2;           //
public float [] AirCO2;     //
public float [] SeaRad;     //
public float [] SeaTemp;    //
public float [] SeaO2;           //
public float [] SeaCO2;     //
public float [] SeaRain;    //
public float [] Risc;       //
public float [] Cost;       //

// ---- Resources ----------
public float [,,] Lres;          // Landscape resource
public float aRes,bRes,maxRes;    // Resources parametre
public float [,,] L;             // (lon,lat,hight)

    // Use this for initialization
    void Start () {
        Initiate();
        StartCoroutine(UpdateState());
    }  // End Start


    void Initiate() {
```

```
        AirTemp =new float [] {0f,0f,0f,0f,0f} ;
        AirRain =new float [] {0f,0f,0f,0f,0f} ;
        AirWind =new float [] {0f,0f,0f,0f,0f} ;
        AirCO2 = new float [] {0f,0f,0f,0f,0f} ;
        SeaTemp =new float [] {0f,0f,0f,0f,0f} ;
        SeaCO2 = new float [] {0f,0f,0f,0f,0f} ;

        // Create a landscape area
        L = new float [100,100,100];
        Lres = new float [100,100,100];
        Lres[10,10,10]=10.0f;

        dTime=1.0f;                     // Start time
        // ---- Set growth resource parametre
        aRes=0.01f; maxRes=1000.0f;
        bRes=aRes/maxRes;
}       // End initiate

void printMe() {
// --- Agent State=X[pos,vel,accel,force,mass,friction,vol,Kost]
          Debug.Log("AbstrLandscape resource: "+Lres[10,10,10]);
}       // End



IEnumerator UpdateState() {
        // --- Curret Agent State ---
        while (true) {
                UpdateLandState();
                printMe();
                dTime = Random.Range(1f, 2f);
                yield return new WaitForSeconds(dTime);
        }  // End while
}       // End dynamic state

void UpdateResourceState() {
        // --- AIR LANDSCAPE ---
```

```
        Lres[10,10,10]=(1.0f+aRes*dTime)*Lres[10,10,10]-bRes*dTime*Lres[10,10,10]*Lres[10,10,10];
            // Cost;    //
    } // End Update Air Temp state

        void UpdateLandState() {
            // --- AIR LANDSCAPE ---
            UpdateResourceState();                          // Update resource state
            //SunRad;       // Sun Radiation
            //AirTemp[0] =aClimateServer.Xatmp[0];// Air temperatur
            //AirRain[0] =aClimateServer.Xrai[0];     // Air raib
            //AirWind[0] =aClimateServer.Xwin[0];   // Air wind
            //AirCO2[0]  =aClimateServer.Xco2[0];   // Air c02
            // --- Sea Landscape ---
            //SeaRad;       //
            //SeaTemp[0] = aClimateServer.Xstmp[0];     // Sea temp
            //SeaCO2[0] = aClimateServer.Xco2[0]; // Sea CO2
            //SeaO2;
            // Risc;    //
            // Cost;    //
    } // End Update Air Temp state

}
```

Agent Server

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class AgentSystem : MonoBehaviour {
//  === FlockingAgent     ===
//  Purpose: Manage an agent flock
//           Create a set of agen populations
//     Compute flocking centre anv velocity
//     Manage a flocking path way
//           Update landscape to agents
//     Processes
//           geneUpdate
```

```
//          stateUpdate
//          landUpdate
//    Input
//          Landscape
//          FlockingTarget
//    Output
//          This to Agents
//    Autors
// Last update 09.10.2009
//    By H. Yndestad
//    ============================

// === The Marine Game ===
//    --- Create landscapes
//    Create sun radiation landscape
//    Create temperature landscape
// Create salinity landscape
//    --- Create genes
//    Plankton genes
//    Zooplankton genes
//    --- Create populations
//    --- Updates
//    Update population state
//    Update landscape
//    Update genes
//    === END GAME ==============

      public TimeServer aTimeServer;

//    === LANDSCAPE ===
//    --- Sea physics ---
      internal float [] myTerrain; // (x,y,z)
      internal float [] seaTemp; // (x,y,z)

// --- Air Physics ---
      internal Vector3 sunRad, sunPos; // (x,y,z)
      internal float [] airTemp; // (x,y,z)
```

```
//      === AGENT ===
        public Agent prefab;
        public List<Agent> agents = new List<Agent>();
        public int AgentId;          // (Type,,,,,,)
        public int AgentType;        // (Type,,,,,,)

// --- Flocking references ---


//      === Flocking Genes ===
//      --- Flocking Id Genes ---
        internal int myId;           // (nr, sex)
        internal int myType;         // (marin, )

// --- Flock Ethics Genes ---
        public Transform target;
        internal float AgentTypeTarget; // (Type,,,,,,)
        internal float AgentPosTarget; // (Type,,,,,,)
        internal float FlockPotentials; // (Speed,Place,,,,,)

// --- Dynamics Genes ---
        internal float dTime,simTime;
        internal Vector3 myPosition, mySpeed, myScale, aPosition, aSpeed;
        public float [] RealTime, SimTime,  EndTime;
        public float waitTime;


//      --- Flocking Genes Dynamics ---
        internal float [] FlockGeneDynamics;
        public float minVelocity = 1;
        public float maxVelocity = 8;
        public int   flockSize=1000;
        public float centerWeight = 1;
        public float velocityWeight = 1;
        public float separationWeight = 1;
        public float followWeight = 1;
        public float randomizeWeight = 1;
        public Vector3 flockCenter;
        public Vector3 flockVelocity;
        public Vector3 center;
```

```
        public Vector3 velocity;

        public Vector3 cameraPos;

// LandscapeSensor(rad, water, salin,soil,,,)
// LandscapeCost(c1,,,,,,,,cn)
        internal float [] Landscapes;    // (Landscape matrix)
        internal float [] sensors;       // (Landscape identification vector)
        internal float [] LandscapeCost;// (Landscape priority)

// ============================================

//GameObject AgentSystem1;

void Start() {
            SimTime =new float [] {0,0,0,0,0} ;       // Start the game
            createGenes();              // Create a set of genes
            createLandscape(); // Update landscape
//          AgentSystem1.transform.position=Vector3(0.0f,0.0f,0.0f);
            GameObject.Find("AgentSystem").transform.rotation =  Quaternion.identity;
            GameObject.Find("AgentSystem").transform.position =  Vector3.zero;
            GameObject.Find("AgentSystem").transform.localScale = Vector3.one;
            StartCoroutine(landUpdate());
            createFlock();        // Create an Angent Flock population
        }     // End start

        void createFlock() {
            // Create an Angent Flock population
            // (By a gene parameter)
            for (int i = 0; i < flockSize; i++){
            // Creates an agent at current position and rotation
                Agent agent = Instantiate(prefab, transform.position, transform.rotation) as Agent;
                agent.transform.parent = transform;                  // Set agent parent position
                agent.transform.localPosition = new Vector3(// Set agent local position
                Random.value * collider.bounds.size.x,        // Set random bound x-siomze
                Random.value * collider.bounds.size.y,        // Set random bound y-size
            // Set random z-size, relative to collider z-size
                Random.value * collider.bounds.size.z) - collider.bounds.extents;
            // --- Set Agent parametre ----------
```

```
            //agent.Aid[1]=(int)(10*Random.Range(0.0f,0.3f));
            //agent.Aid[2]=(int)(10*Random.Range(0.0f,0.2f));
            agent.controller = this;                    // Set this data to agent controller
            createGenes();
            agents.Add(agent);                          // Set agent to agent list
        }   // End For
}   // End create Agent flock

void createGenes() {
        // Create a sett of agent genes
        //int AgentId=1;
        //float flockSize = Random.Range(10f, 90f);
        // Set genes to the agent list
}   // End start

void createLandscape() {
        // Create a sett of agent genes
        // Set genes to the agent list
}   // End start

void UpdateTime(){
        //simTime=aTimeServer.SimTime[1];
        simTime=Time.realtimeSinceStartup;
        //float simTime=aTimeServer.SimTime[0];
        //float waitTime = Random.Range(0.2f, 1.5f);
        waitTime = 5f;
        //print("simTime = " +simTime);
}  // End Update


IEnumerator landUpdate() {
        // Read new landscape state
        // Set new landscape to agents
        while (true) {
            UpdateFlock();                              // Update flock state
            UpdateSunState();                       // Update Suns States
            UpdateTime();
```

```
                yield return new WaitForSeconds(waitTime);
            }   // End while true
        }   // End landUpdate

        void UpdateCameraPos(){
            cameraPos=sunPos;
            //print("Sun Rad = " +sunRad);
        } // End Update

        void UpdateSunState(){
            Vector3 sunRad = new Vector3(20,20,20);
            sunPos=new Vector3(500,50,200);
            sunRad.y= 20+20*Mathf.Sin(2*3.14f*simTime/(60));
            //print("Sun Rad = " +sunRad);
        } // End Update

        void UpdateFlock(){
            // Update Flock center and velocity
            foreach (Agent agent in agents){
                center += agent.transform.localPosition;
                velocity += agent.rigidbody.velocity;
                } // Updates center and velocity
            flockCenter = center / flockSize;
            flockVelocity = velocity / flockSize;
        } // Update Flock

} // End Program
```

Agent

```
using UnityEngine;
using System.Collections;
//using System.Collections.Generic;

public class Agent : MonoBehaviour {
//      === Agent Game ============
//      --- Agent 1 Etichs ---
//              1. Landscape (Terranin, Sun Radiation, Flock)
//              2. Cost (+/-Terrain, - Sun Radiation, +/-Flock, spawning)
```

```
//              2. Growth(+/- Cost, growthrate)
//              3. Recruitmen (agent sex, Cost, spawning)
//      Agent 1 Learning
//              1. Position (Recruitment) (Random by landscape)
//              2. Growth (Position, cost)(Genes)
//              3. LifeCycle(spawning, mortality)(Genes)

// --- Agent 2 Ethics ---
//              1. Landscape (Sea volume, Agent 1)
//              2. Cost (+Movement, -Target +spawning)
//      Agent Learning
//              1. Position(Target tracing)
//              2. Growth(cost)(genes)
//              3. LofeCycle(spawning, morttality)(gene)

//      Input
//              AgentSystem(Landscape, agentState, flockingState)
//              TimeServer (TimeState)

//      Output
//              LandScape (agentState)

//  Project: Virtual More
//      Autor: H. Yndestad
//  Last update 17.11.2009
//      ============================

// --- Array dimension ----------
    internal int dim=9;                                 // Array dimensions
// --- Time data ---
    internal float dT;                                  // Delta time

// ==== Agent Format ===========
// --- Aid = agent[nr, type, sex, spawn, value]
// --- Agent State=X[pos,vel,accel,force,mass,friction,vol,colour]
// --- Agent Force=F[sum,gravity,friction]
        public int [] Aid = new int[9];             // Agent identity vector
    internal Vector3 [] X=new Vector3[9];  // Agent state vector
    internal Vector3 [] Xp=new Vector3[9];      // Past state vector
```

```
    internal Vector3 [] Xe=new Vector3[9];       // Error state vector
        internal Vector3 [] F=new Vector3[9];      // Error state vector

// ====== Landscapes Format ===========
// Lph = physics[terrain, sunrad, temp]
// Lfl = flock[mean, aligment, sparation]
// Lab = abstract[type, pos, kost]
// Lag = agent[type, pos]

// --- Landscape=[ter,sunPos,sunRad]
    internal Vector3 [] L=new Vector3[9];  // Landscape vectors
    internal Vector3 [] Lp=new Vector3[9];       // Past Landscape vector
        internal Vector3 [] Lph=new Vector3[9]; // Ag Landscape
        internal Vector3 [] Lfl=new Vector3[9]; // Flocking Landscape
    internal Vector3 Vel=new Vector3();
// --- Targets ---
// Tph = physics[terrain, sunrad, temp]
// Tab = abstract[type, pos, kost]
// Tag = agent[type, pos]

// Target=[position]
    internal Vector3 [] T=new Vector3[9];  // Target Vector
        internal Vector3 [] Tp=new Vector3[9];    // Paste Target Agent data
        internal Vector3 [] Tab=new Vector3[9]; // Abstract Landscape Target
        internal Vector3 [] Ta=new Vector3[9];    // Agent Target
        internal Vector3 [] Tph=new Vector3[9]; // Physics Tareget

// --- Id ---
// Apo = position[pos, velos, accel, force]
// Abo = body[energy, mass, vol, friction, colour]

// === Genes ======================================
// Gtr = tracing[Leanrate]
// Gre = recruit[maturity, recruit, mortality]
// Genes=[posRate, growRate, flockAligmRate, flockSepRate]
        internal Vector3 [] G=new Vector3[9];     // Genetic Agent vectors
        internal Vector3 [] Gp=new Vector3[9];   // Past Gene vector state
        internal float []   Gfl=new float[9];  // Flocing Gene vectors
```

```
        internal float []   Gtr=new float[9];   // Tracking Gene vectors
        internal float []   Gre=new float[9];   // Recruit Gene vectors

// === FLOCKING FORMAT ================================
// Flocking = [flockTargetPos, flockMeanPos,flockVeloc, relativPos, separation]
    internal Vector3 [] Fl=new Vector3[9];        // Flocking vector state
        internal Vector3 [] Fp=new Vector3[9];   // Flocking vector state
        internal Vector3 g=new Vector3();         // Gravity force vector

//  === COST FUNCTIONS ==========================
        internal Vector3 []  J=new Vector3[9];    // Agent object function vector
        internal Vector3 []  Q=new Vector3[9];    // Agent cost vector
        internal Vector3 []  K=new Vector3[9];    // Agent force gain kontrol

        //public Transform target;       // Flocking references


    // ====================================================
        public TerrainData terr;
        internal AgentSystem controller; // AgentSystem copy

        void Start() {
                SetMyIdentity();
                SetMyParametre();
                StartCoroutine(updateState());
                //setMyGenes();
        }       // End Start

        void SetMyIdentity() {
                // Aid = agent[nr, type, Target]
                //Aid[0]=   ;
                //Aid[1]= Type: , , , , ;
                //Aid[2]= Target:0=Land position, 1=Sun, 2= Moveing target;
                Aid=new int[dim];// Id vector: {Nr, type,sex}
                Aid[1]=(int)(10*Random.Range(0.0f,0.3f));
                Aid[2]=(int)(10*Random.Range(0.0f,0.2f));
        }       // Set my identity
```

```
void SetMyParametre() {
	for (int i=0; i<dim; i++) {
		X[i] = 10.0f*(new Vector3(Random.value,Random.value,Random.value));
		Xp[i] = new Vector3(Random.value,Random.value,Random.value);
		L[i] = new Vector3(Random.value,Random.value,Random.value);
		Lp[i] = new Vector3(Random.value,Random.value,Random.value);
		T[i] = new Vector3(Random.value,Random.value,Random.value);
		Tp[i] = new Vector3(Random.value,Random.value,Random.value);
		G[i] = new Vector3(Random.value,Random.value,Random.value);
		Gp[i] = new Vector3(Random.value,Random.value,Random.value);
		F[i] = new Vector3(Random.value,Random.value,Random.value);
		Fp[i] = new Vector3(Random.value,Random.value,Random.value);
		// Gtr = tracing[Leanrate, ContrGain]
		Gtr[i]=Random.value;	// Traccing genew
		Gfl[i]=Random.value;	// Flocking gene vector
		// Gre = recruit[maturity, recruit, mortality]
		Gre[i]=Random.value;	// Recruitment gene
		// Set control gene vector
		// Set cost gene vector
	} // End for
	// Gravity force
	//F[1]=Vector3.zero;
	F[1].y=-9.81f;  // F=-g*M
}	// End target identification

void updateGenes() {
	// Update genes when mature
	// Crossing genes wwith the best
	// Mutate genes
}	// End

void updateLifeCycleState() {
	// Update genes when mature
	// Crossing genes wwith the best
	// Mutate genes
}	// End
```

```
    void printMe() {
        // --- Agent State=X[pos,vel,accel,force,mass,friction,vol,Kost]
        Debug.Log("I= "+Aid[1]+" X= "+ X[0]+" V= "+X[1]+" A= "+X[2]+" F= "+X[3]+" M= "+X[4]+" Fric="+X[5]+" J=
"+X[7]);
    }   // End

    IEnumerator updateState() {
        // --- Curret Agent State ---
        while (true) {
            dT=Time.deltaTime;                  // Get time change
            ReadTarget();                       // Update landscape state
            IdentAgent();                       // Update agent state
            flockingControl();                  // Update flocking state
            trackTarget();                      // Compute agent control
            updateCost();                       // Update cost function
            //printMe();

            // dT etter samplingsteoremet og avstand
            float waitTime = Random.Range(0.2f, 1.5f);
            yield return new WaitForSeconds(waitTime);
        }  // End while
    }   // End dynamic state


    void ReadTarget() {
        // === Landscapes ===
        getLandscape(); // Read terrain position
    // --- Sun Landscape ---
        L[1]=controller.sunPos;         // Sun Position
        //Lph[1]=new Vector3(500,50,200);
        L[2]=controller.sunRad;             // Sun Radiation
        // --- Targets ---
        Ta[1]=controller.target.localPosition;      // Position of a moving target
    }   // End Indent Landscape

    void getLandscape() {
        int ySize=600;
```

```
        L[0]=new Vector3(325f,1f,200f); // Set position
        // Read terrain hight
        L[0].y = terr.GetHeights(325,200,1,1)[0,0] * ySize;
//print("L[0] "+ L[0]);
}       // End Indent Landscape


void IdentAgent() {
        // === Agent states ===
        // Apo = position[pos, velos, accel]
        //Apo[0]=transform.localPosition;    // Get agent old position
        //Apo[1]=rigidbody.velocity;                    // Get agent old velocity
        X[0]=transform.localPosition;      // Agent New position
        //X[1]=rigidbody.velocity;         // Agent New velocity
}       // End Ident Agent



void trackTarget() {
        // === Tracing Algorithm ============
        // --- Target selection ---
        float Lr=Gtr[0];                // Learning rate gene
        Vector3 R= new Vector3();
        if (Aid[1]==0) {
                R=L[0]; // Ground Target
        }  // End Ground target

        if (Aid[1]==1) {
                R=L[2]; // Sun radiation
                //Debug.Log("Id= "+Aid[1]+" J= "+ X[8]);
        }  // Ende sun position

        if (Aid[1]==2) {
                R=Ta[1]; // Target
        }  // End select target

// === Target Tracking Method ===
```

```
        // --- Agent State=X[pos,vel,accel,force,mass,friction,vol,Kost]
        // Tracking algorithm
        // Disanxe Error:      E=(T-X)
        // Force Control:      F=K*E-KvV
        // Accelration:        A=F/M + g
        // Speed:              V=V+A = V + F/M
        // Distance Target Error
        Xe[0]=Lr*Xe[0]+(1-Lr)*(R+Lfl[0]-X[0]); // Estimated force
        float K=1.0f;           // Control gain
        F[0]=Xe[0]*K;           // Force gain
        float Kv=X[5].y*0.1f;           // Resist coef
        F[0]=F[0]-X[1]*Kv;   // Resistance force
        // Gravity
        Vector3 Fg=new Vector3(0,-9.81f,0);
        float M=X[4].y;
        F[0]=F[0]+0.1f*Fg*M;            // Add gravity
        // Force=Ferror+Fgravity-Fresist
        //Vel=Lr*Vel+(1-Lr)*F[0]/M;
        //rigidbody.velocity=Vel;
        rigidbody.velocity=F[0];
}  // End agent


void updateCost() {
// --- Agent State=X[pos,vel,accel,force,mass,friction,vol,Kost]
// === Estimate Movement Force ===
        // V(n)=(X(n)-X(n-1))/dT
        // A(n)=(V(n)-X(n-1))/dT
        // F(n)=M*A(n)
         X[1]=(X[0]-Xp[0])/dT;      // V(n)=(x(n)-x(n-1))/dT
         X[2]=(X[1]-Xp[1])/dT;      // a(n)=(v(n)-v(n-1))/dT
         X[3]=X[2]*X[4].y;          // F(n)=a(n)*m
         // Friction forvce=
         // Gravity force =

// --- Estimate Cost function ---
        // Object fubction
```

```
        // J = XQXナス+ LPLナス

         float le=0.95f;
         X[7]=le*X[7]+(1-le)*X[3]*dT;          // J(n)=J(n)+F(n)*dT
        // J = J + Q*Fm*dT + Q*L
        // Update energy input from target
        // E = R*T
        Xp=X;      // Update past state
}       // End


void flockingControl() {
        // === Separation: Agent separation distance ===
        Vector3 separation = Vector3.zero;
        foreach (Agent agent in controller.agents) {
                if (agent == this) {    // To all agents exept to this agent

                }       // End if
                else {
                        // Compute distanse to other agents
                        Vector3 relativePos= transform.localPosition-agent.transform.localPosition;
                        // Accumulate separation index separation = relpos/sqr(relpos)
                        separation += relativePos/(relativePos.sqrMagnitude);
                } // End else
        }       // End foreach

        // === Flock target identifications ===
        //Lf[1]=controller.target.localPosition;     // Position of a moving target
        Lfl[1]=controller.flockVelocity;              // Mean flock velocity
        //Lfl[2]=separation;                                    // Separation in speed
        Lfl[2]=separation*dT;                                   // Separation in position

        // === Flocking Gene Control ====
        Lfl[0]=Gfl[1]*Lfl[1]+Gfl[2]*Lfl[2];
        //totalFlockDist=Lf[0];
//print(Lfl[0]);
        // --- Follew flocking target
```

```
    }// End flocupdate

}   // End Agent
```