AIMC 2023

Revisiting Reynolds - Autonomous Agents for Spatial Audiovisual Composition and Performances

Damian Dziwis[1,2]

[1]TH Köln - University of Applied Sciences, Cologne, Germany,
[2]Technische Universität Berlin, Germany

Published on: Aug 29, 2023

URL: https://aimc2023.pubpub.org/pub/t2bsu95w

ABSTRACT

Autonomous agents represent a special class of systems in the context of Artificial Intelligence (AI) and Artificial Life. As embodied AI, they are systems that have a certain understanding of their environment and adapt their behavior accordingly. They are used in a wide range of fields, from robotics to video games, but also in artistic practice. Craig W. Reynolds' work on steering behavior and flocking simulation provided an important framework for simulating the motion of autonomous agents. Originally intended for use in computer games, various adaptations and inspired applications can be found in a wide variety of domains, including spatial composition. The motion of autonomous agents in 3D space can be used to control spatial sound sources and other virtual objects to realize life-like behavior in improvisations or conducted performances. In this paper, we describe the implementation of steering behaviors proposed by Reynolds as autonomous agents in the modular 3D engine IVES for the Max development environment. Furthermore, we demonstrate the potential to realize spatial audiovisual compositions and performances with the improvisational behavior of autonomous agents in combination with a 3D engine specialized for art production.

Keywords: Autonomous Agents, Audiovisual Composition, AI Improvisation, AI Performance, Artificial Life, Embodied AI

Introduction

Movement, even outside the performing arts such as dance, is an essential element throughout many domains of artistic practice. In the field of spatial composition, the creation of sound source trajectories is an important technique for electroacoustic and acousmatic compositions [1]. Already since the 1950s, composers such as Edgard Varèse with "Poème électronique" (1958) in the Philips Pavilion designed by Iannis Xenakis, have been exploring techniques and technologies for the spatial arrangement and movement of sounds through loudspeakers in space. As loudspeaker-based spatial audio evolved from stereophony for 1D or 2D spaces to 3D sound field reproduction with technologies such as Ambisonics or Wave Field Synthesis [1], the dimensions and possibilities of sound movement for spatial compositions also expanded. To cope with these extended possibilities, also a variety of software emerged to support the creation of complex trajectories and spatial compositions. Software with audio processing and complex trajectory editors, such as Zirkonomium [2], Holophon [3] or Sound Trajectory 2 [4], allow not only the playback of audio material for loudspeaker arrays, but also the design of accurate three-dimensional trajectories using graphical user interfaces (GUIs). Other approaches, such as the graphical sequencer Iannix [5], do not involve direct audio processing, but allow, among other things, the creation of multidimensional trajectories. This can be done both graphically and by implementing generative algorithms in the programming language JavaScript. In addition to graphical editors, such algorithms are another key method for generating trajectories and movement of sounds in the context of spatial composition. These algorithms can range from aleatoric/random processes, generative construction of complex geometric figures and shapes, to the simulation of motion based on various models. The Spat programming library [6] for the Max development environment [7] provides objects for spatial audio processing as well as algorithms for trajectory and motion generation. In addition to trajectory generation algorithms based on 2D and 3D geometric shapes, Spat also includes objects with autonomous steering behaviors, like the simulation for motion (simone) object [8], or a flocking simulation (boids) based on the Reynolds boids concept [9]. Starting in the 1980s, Craig Reynolds' research made important contributions to the simulation of flocking and herding, and to the steering behavior of autonomous agents as characters [10]. Autonomous Agents are an approach in Artificial Intelligence (AI) and the related field of Artificial Life [11]. As embodied AI [12], they are characterized as autonomous systems that have (limited) knowledge of their environment and adapt their behavior accordingly. According to Stan Franklin's definition, an autonomous agent "is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future" [13].

Autonomous agents can be implemented, for example, as systems that behave in a lifelike and improvisational way among themselves or by navigating through their environment [10]. In this way, virtual characters and objects can be realized that autonomously show plausible behavior in specific situations. Although Reynolds' research originated in the context of virtual environments and computer games, his algorithms for steering and flocking behavior have been used in a variety of domains, including spatial composition. Game engines such as Epic's Unreal Engine [14] or Unity [15] are thus possible applications for autonomous agents in games and virtual worlds. Besides their main purpose of game development, they also offer the possibility of spatial audio and composition, especially in the context of spatial, audiovisual, and virtual art. In contrast to the game engines mentioned above, the modular 3D engine IVES [16] is not intended for game development, but specifically for use in research and art production, offering advanced possibilities for spatial audio and generative audiovisual design. As a toolkit with open-source modules for the Max programming environment, IVES applies the Spat library for spatial audio processing and the OpenGL-based Jitter library integrated in Max for visual rendering of 3D elements. IVES provides a wide range of modules for spatial audio rendering using loudspeaker arrays via Ambisonics as well as headphone-based rendering via binaural reproduction [17]. 3D objects and entire virtual environments can be created for screen or projection use, as well as for common virtual reality (VR) systems. Since IVES is implemented as modules for Max, the Max programming language can furthermore be used to realize arbitrary sounds and visual objects, using generative algorithms or media data (e.g. audio files, 3D models, etc.).

Although IVES already has the trajectory algorithms from the Spat library [18] integrated as a module, this paper describes the effort to develop modules for autonomous agents. Unlike static trajectories, which specify a fixed path for sound sources or virtual objects, the autonomous agent modules can be used to implement dynamic behavior in the form of movement relative to the user or to other agents or objects. In this way, improvised performances can be realized or spatial composition conducted. The integration into IVES makes it possible to realize them not only for spatial audio reproduction via loudspeakers or headphones, but also as audiovisual virtual artworks for screens, stages or VR. Since the steering behaviors proposed by Reynolds are easy to implement, computationally efficient, and yet can be quite effective, especially in creative applications, the goal of this work is to integrate them as modules for autonomous agents in IVES. The following sections describe the underlying steering algorithms, their implementation as autonomous agents, and their integration as modules within the IVES engine. Furthermore, use cases for possible artistic application in virtual audiovisual works are demonstrated.

Autonomous Steering Behavior

The scope of this work is within the framework of Reynolds's "Steering Behaviors For Autonomous Characters" [10]. This section provides a summary of Reynolds' approach and the steering behaviors used for autonomous agents in IVES. In his paper, Reynolds describes algorithms for several steering behaviors in autonomous characters, defined as autonomous agents "intended for use in computer animation and interactive media such as games and virtual reality” ([10], p. 1). Following the taxonomy outlined by Reynolds, these autonomous agents can be understood as situated, embodied, reactive and virtual agents. The proposed algorithms aim to steer these agents' behavior according to a simple "vehicle" model [19]. ‘Behavior’, in the context of these autonomous agents, is here defined in a top-down hierarchy of "Action Selection: strategy, goals, planning" -> "Steering: path determination" -> "Locomotion: animation, articulation" ([10], p. 2). In Reynolds’s approach, the emphasis is on ‘Steering’, which is implemented in the form of steering forces, that can also be combined, to provide lifelike, improvisational movement between each other and the environment. ‘Locomotion’ is applied to a simple vehicle model using point mass approximation. It is defined by the properties of mass, position, and velocity, with the latter two being vectors. To generate motion, for each frame steering forces are applied to the vehicle's point mass in relation to a given target. The ‘Action’ behavior of the autonomous agents is not determined by a higher-level goal or strategy, but only by the motion of a given target. As a result, the agent's movement is incrementally adapted by the applied steering behavior in relation to that target.
The steering forces described by Reynolds include: seek, flee, pursuit, evasion, offset pursuit, arrival, obstacle avoidance, wander, path following, wall following, containment, flow field following, unaligned collision avoidance, separation, cohesion, alignment, flocking, and leader following. Subsequently, only those steering forces are summarized that have been implemented as autonomous agents for IVES in the context of this work.

Seek (flee, arrival, pursuit, evasion): The seek steering force moves the agent to a particular position by directing the velocity radially towards the target. By forming a "desired velocity" ([10], p. 9) vector with the direction from the agent to the target, and with the length of the agent's speed property, a steering velocity can be calculated by subtracting this desired velocity from the agent's current velocity. The following steering forces are variations of the seek force. Flee behavior is the inversion of seek force, in which the agent's velocity is directed radially away from the target by applying the desired velocity in the opposite direction. Using seek force, the agent will cross the target as soon as it reaches it and then turn around to reapproach it again. To stop the agent once it reaches the target position, the "arrival" steering force extends the seek behavior by linearly decreasing the agent's desired velocity after it approaches closer than a specified radius until it equals zero at

the target position.
Pursuit is used to seek another autonomous agent. The pursuit force is also based on the seek behavior but extends it by predicting the target's future position. The future position of the target agent can be predicted by multiplying the velocity of the target by a specified factor, and adding the result to the current position. The predicted future position is then the target position seeked by the pursuing agent. Evasion is again the inverse of pursuit behavior, using the flee force to evade the target agent and its predicted future position in the opposite direction.

Wander: Wander is a steering force that does not act in relation to a given target, but instead steers the agent through the environment in a certain random manner. This is done by limiting the random steering force to the surface of a sphere in front of the agent. At each frame, a random displacement is added to the previous steering force, with the sum also being limited to the surface of the sphere. The degree of direction is thereby determined by the radius of the sphere, while the magnitude of the random displacement determines the rate of the random changes.

Path Following: Path following allows steering an agent along a given path. Instead of moving strictly according to this path, the steering behavior enables the agent to move along the path with deviations within a certain radius. This creates a more lifelike and improvisational movement compared to predefined trajectories. Path following is realized by making a prediction of the agent's future position and projecting that predicted position to the nearest point on the path. If the distance from this predicted position to the nearest point is smaller than the specified radius, then the agent is already following the path correctly and does not need to be corrected. Otherwise, the seek behavior is used to steer towards the projection of the predicted future position on the path.

Flocking: Flocking is a combined behavior applied to a group of autonomous agents. It was introduced as the boids model in 1987 by Reynolds in his paper "Flocks, Herds, and Schools: A Distributed Behavioral Model" [9]. Here Reynolds first introduced his approach to simulating the swarm behavior of multiple agents, which he calls boids, an abbreviation for "bird-oid". The simulation is realized with three combined steering behaviors: **separation**, the ability to keep a distance from other nearby agents; **cohesion**, the ability to group with other nearby agents; and **alignment**, the ability to align in direction and speed with other nearby agents. These three steering forces can be simply added together, or beforehand normalized and scaled by a weighting factor for better control. Thus, the flocking behavior, for each of the three steering forces, can be defined by the parameters weight, as well as distance and angle to define the neighborhood.

Implementation

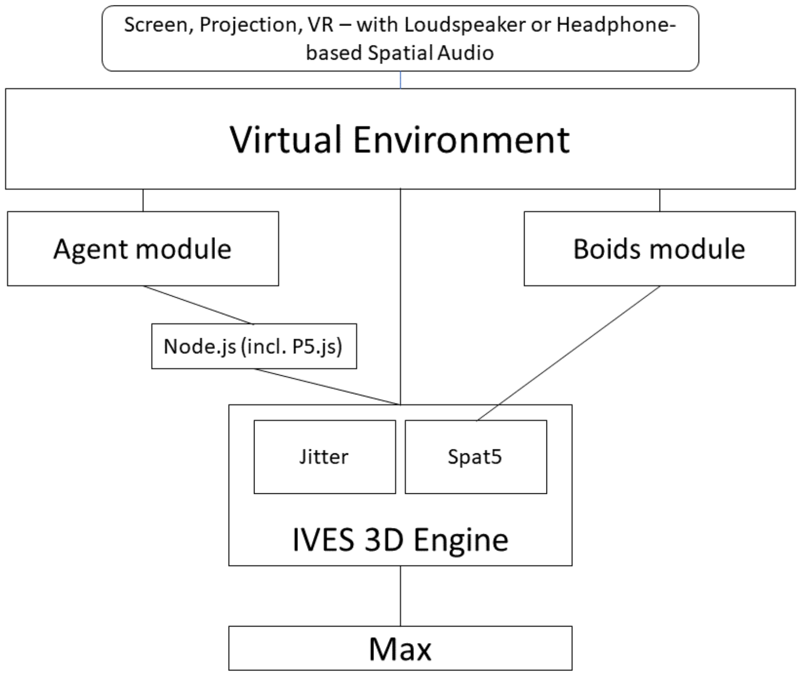


Figure 1: a diagram showing the architecture of the two autonomous agent modules integrated in IVES.

In this work, the steering behaviors proposed by Reynolds were implemented as two modules for the IVES engine. They were developed in the Max programming environment using the Node.js and P5.js [20] JavaScript frameworks (Agent module) and the Spat library (Boids module) (see Fig. 1).

Agent Module:

The implementation of the above-mentioned steering algorithms as an IVES Agent module is based on Daniel Shiffman's implementation of the steering behaviors proposed by Reynolds. In "The Nature of Code" [21] and "The Nature of Code 2" [22] Shiffman describes his implementations in Java (using the Processing framework [23]) and JavaScript (using the P5js framework [20]) with corresponding code examples. The code published by Shiffman has been adapted for use in Max and the IVES 3D engine: Shiffman's 2D implementation has been extended to 3D and reimplemented in Max's Node.js environment (including the port of Java code). While the P5js framework is still integrated to use the included vector functions, the rendering pipeline of P5js is not used, and instead, the code is adapted and integrated into the IVES rendering and object handling.
The control behaviors implemented as an IVES Agent module are: seek, flee, arrive, pursue, evade, wander, and path following.
Following Shiffman's Vehicle class, the core of the Agent module is the class 'Agent' with the attributes: acceleration, velocity, position, maxspeed, maxforce, radius, distance, and id:

```
class Agent{
    constructor(x, y, z){
         this.id = 0;
         this.acceleration = new Vector(0, 0, 0);
         this.velocity = new Vector(0, 0, 0);
         this.position = new Vector(x, y, z);
         this.maxspeed = 2;
         this.maxforce = 0.3;
         this.radius = 1;
         this.distance = 5;
         [...]
    }
```

Code 1: Constructor of the Agent class with the relevant attributes describing the agent's behavior.

While the id attribute is used for object management within the IVES engine, the other attributes are relevant for the implemented steering behavior algorithms: compared to Reynolds Simple Vehicle Model -

```
Simple Vehicle Model:
     mass scalar
     position vector
     velocity vector
     max_force scalar
     max_speed scalar
     orientation N basis vectors
```

Code 2: Reynolds Vehicle Model for Autonomous Characters ([10], p. 6) and its proposed attributes.

we can see that the mass and orientation attributes are missing, but acceleration, radius, and distance attributes have been added instead. Analogous to Shiffman, also our physical model does not have a defined mass. Furthermore, in Shiffman's implementations, as in the present adaptation, the steering behaviors are implemented as methods of the main class. Here Shiffman implements the steering forces as a single return vector, "force", of the behavior methods. This is done according to Reynolds's concept of a desired vector and steering vector:

```
desired_velocity = normalize (position - target) * max_speed
steering = desired_velocity – velocity
```

Code 3: Reynolds force concept for Autonomous Characters ([10], p. 9).

This concept, and Shiffman's proposed implementation, was extended and adapted for use in 3D and IVES. At each frame, the force vector is calculated by the selected behavior method, and the result is added to the agent's acceleration attribute. In a subsequent update method, the agent's velocity is summed up with its acceleration, the magnitude is limited to the maxspeed attribute, and finally, the agent's position is added with the calculated velocity to produce its new position. The acceleration attribute is then reset for the next iteration:

```
update(){
     this.velocity.add(this.acceleration);
     this.velocity.limit(this.maxspeed);
     this.position.add(this.velocity);
     Max.outlet("/agent/vec "+this.id+" "+this.position.x+" "
     +this.position.y+" "+this.position.z+" "+this.velocity.x+" "
     +this.velocity.y+" "+this.velocity.z);
     this.acceleration.mult(0);
}
```

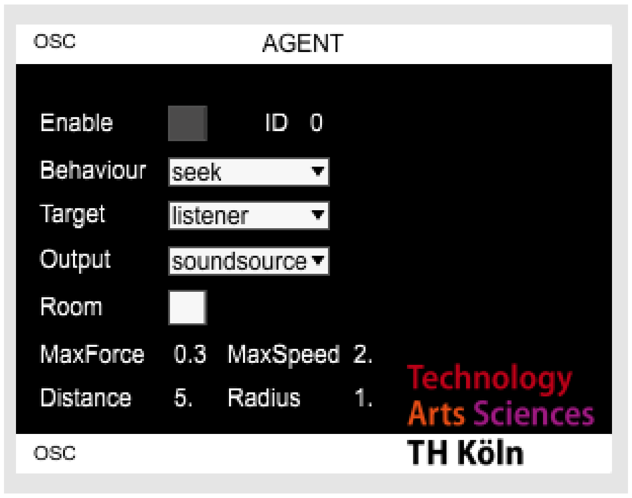
Code 4: update method for autonomous agents as proposed by Shiffman, adapted to use in IVES.

As in Shiffman, this is followed by a method that renders the agent and its changes in position. In this method, the agent is rotated with respect to its velocity vector, eliminating the need to keep its orientation as an attribute as proposed by Reynolds. For use in IVES, both position and rotation are output as OSC messages to control linked IVES modules such as sound sources or 3D objects:

```
move(){
   let rho = this.velocity.mag();
   let phi = Math.acos(this.velocity.z/rho);
   let theta = Math.atan2(-this.velocity.y, this.velocity.x);
   if(output == 0){
     Max.outlet("/source/"+this.id+"/xyz "+this.position.x+" "
     +(this.position.z * -1)+" "+this.position.y);
     }else{
     Max.outlet("/shape/pos/x "+this.position.x);
     Max.outlet("/shape/pos/y "+this.position.y);
     Max.outlet("/shape/pos/z "+this.position.z);
     Max.outlet("/shape/rot/y "+ phi * (180.0 / Math.PI));
     Max.outlet("/shape/rot/x "+ (theta * (180.0 / Math.PI)));
     }
}
```

Code 5: method to move and rotate an arbitrary IVES element (e.q. sound source or 3D object), according to the calculated agent's position and velocity.

According to the described procedure, all listed steering behaviors have been implemented. The remaining attributes radius and distance are used to parameterize the arrival and wandering behavior. These attributes, as well as the desired steering behavior, can be parameterized in the graphical user interface (GUI) of the IVES Agent module (see Fig. 2). Here it is also possible to choose the target between the listener (i.e. the interactive first-person user in the virtual environment), another linked agent, or a virtual object. The output object, which is steered by the behavior of the autonomous agent, can be chosen between sound sources or other virtual objects.


OSC AGENT

Enable ID 0

Behaviour seek

Target listener

Output soundsource

Room

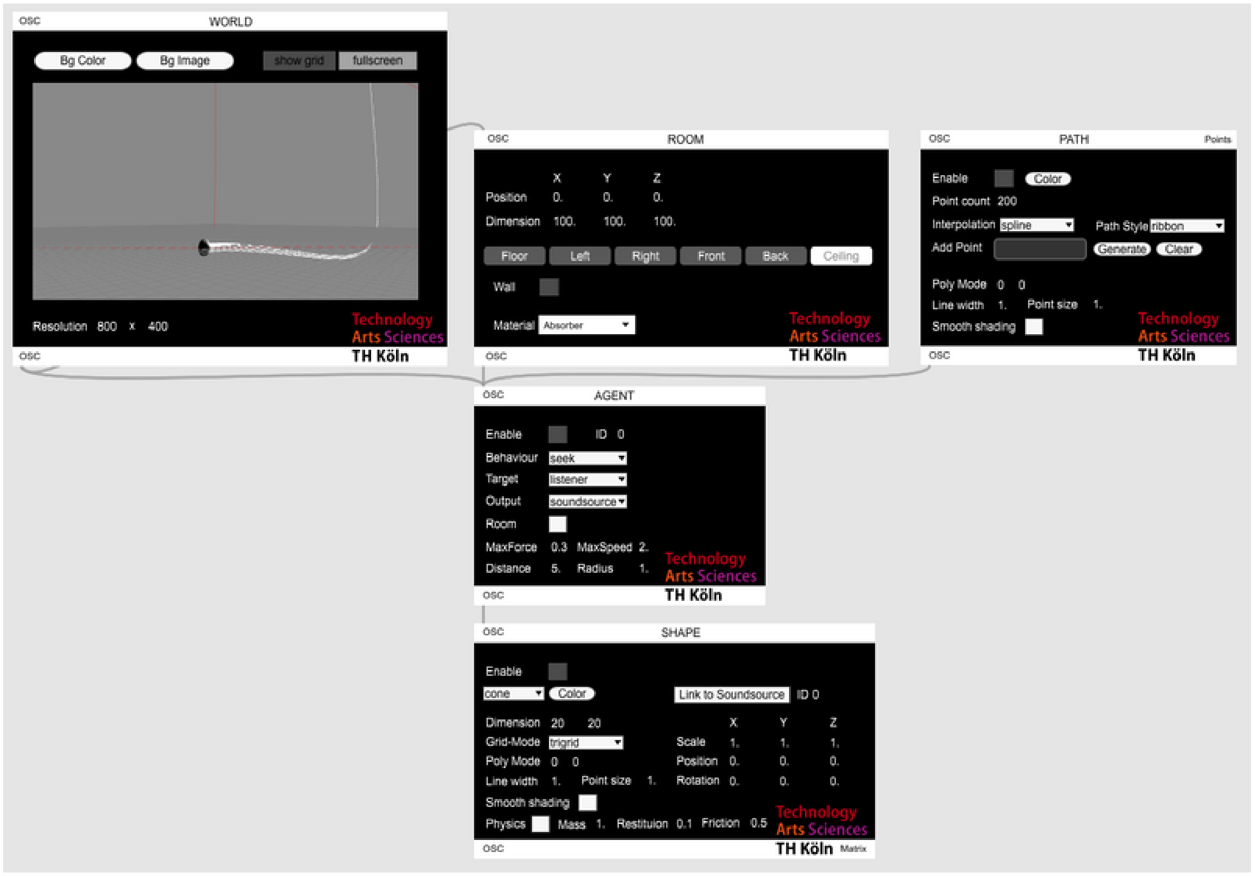
MaxForce 0.3 MaxSpeed 2.

Distance 5. Radius 1.

OSC

Figure 2: a screen capture showing the Agent module.

In addition to interacting with virtual objects as target and output elements, the Agent module can also integrate with other modules of the IVES engine (see Fig. 3). The "Boundaries" behavior, proposed by Shiffman as a steering force to constrain agents within a given space [24], has been ported and extended for combination with the IVES Room module. Originally developed to define acoustic spaces for spatial room simulation[25], the Room module also allows the definition of a three-dimensional room that serves as a boundary to limit the action space of an agent when the room parameter is selected. The IVES Path module allows the definition of three-dimensional paths that can be traversed by agents using the path-following behavior. All parameters such as steering behaviors, attributes, targets, and output elements can also be controlled and automated for compositions or performances using OSC messages, like with the IVES Animate module.



WORLD: Resolution 800 x 400

ROOM:

	X	Y	Z
Position	0.	0.	0.
Dimension	100.	100.	100.

Material Absorber

PATH: Point count 200; Interpolation spline; Path Style ribbon; Poly Mode 0 0; Line width 1.; Point size 1.

AGENT: ID 0; Behaviour seek; Target listener; Output soundsource; MaxForce 0.3; MaxSpeed 2.; Distance 5.; Radius 1.

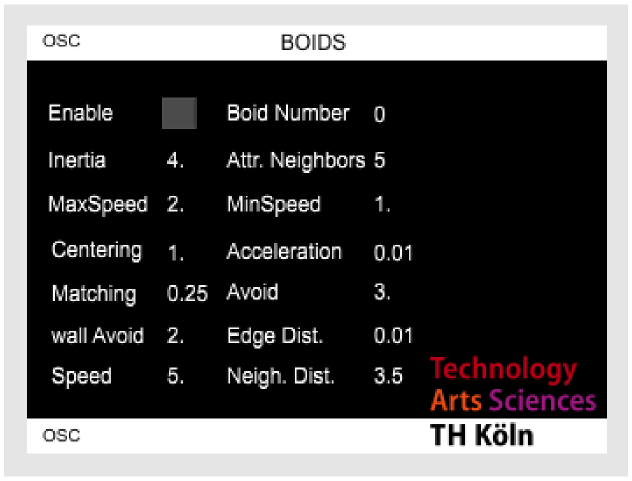
SHAPE: cone; ID 0; Dimension 20 20; Grid-Mode trigrid; Poly Mode 0 0; Line width 1.; Point size 1.; Mass 1.; Restitution 0.1; Friction 0.5

	X	Y	Z
Scale	1.	1.	1.
Position	0.	0.	0.
Rotation	0.	0.	0.

Figure 3: a screen capture showing a Max patch using the Agent module with the IVES Room, Path and Shape module to create a scenery in a virtual environment.

Boids Module

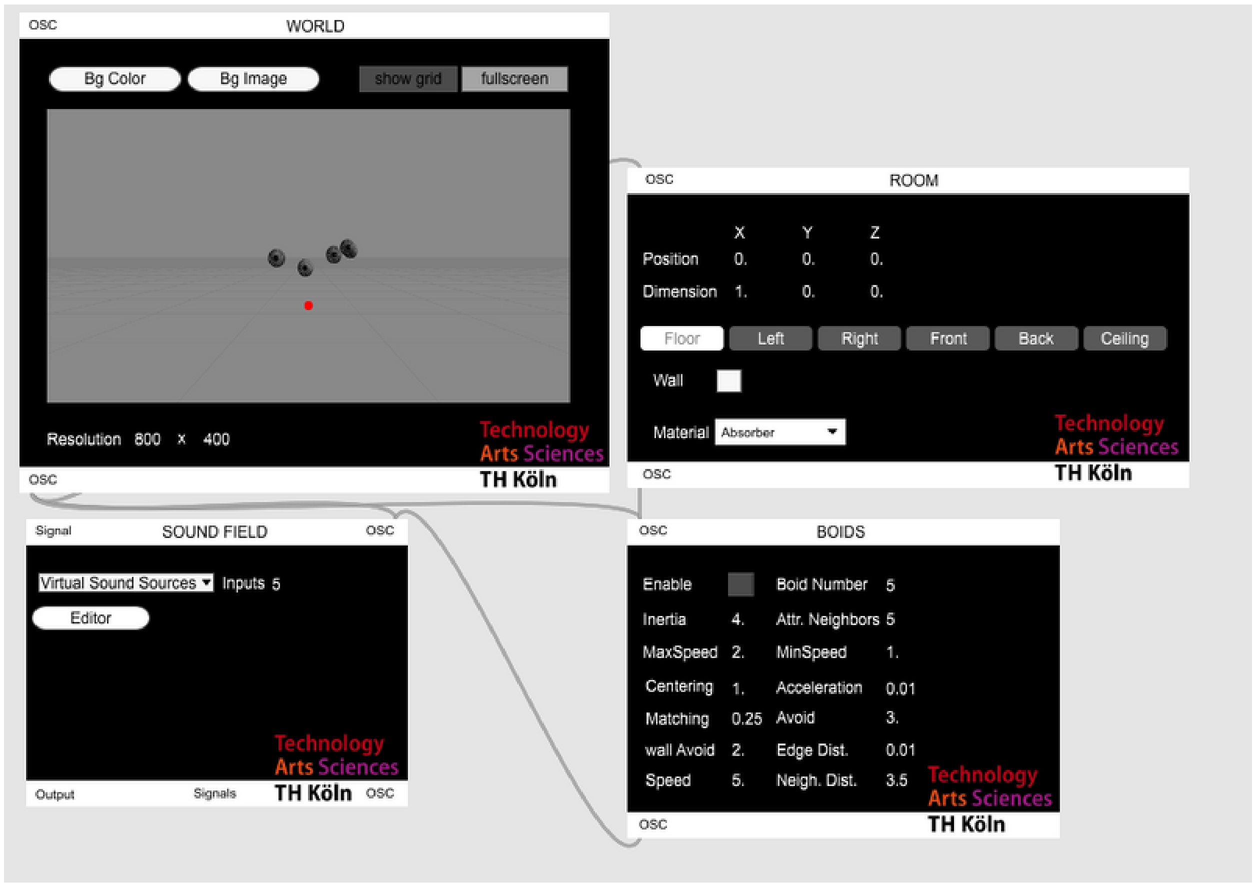
Since it is inconvenient in practice to implement flocking behavior as a cascade of multiple Agent modules, this group behavior was implemented as a separate module. Unlike the agent module, the Boids module is not based on Shiffman's implementation. Instead, since the Spat library integrated in IVES already includes an object with an integrated flocking algorithm for boids, this implementation was used as the basis for the Boids module. The Spat boids algorithm allows extensive parameterization of the integrated steering behaviors for separation, cohesion, and alignment, as well as extended behaviors of the individual agents. The Spat boids object has been implemented as a module together with a GUI for the behavior parameters and also allows to control whole groups of sound sources or virtual objects in IVES (see Fig. 4).



Enable ■ Boid Number 0
Inertia 4. Attr. Neighbors 5
MaxSpeed 2. MinSpeed 1.
Centering 1. Acceleration 0.01
Matching 0.25 Avoid 3.
wall Avoid 2. Edge Dist. 0.01
Speed 5. Neigh. Dist. 3.5

Figure 4: a screen capture showing the Boids module.

Again, the Boids module can interact with the IVES Room module to define the bounding box parameter for the boids object, the walls within which the boids can move, by creating a three-dimensional room (see Fig. 5).



Resolution 800 x 400

	X	Y	Z
Position	0.	0.	0.
Dimension	1.	0.	0.

Wall □
Material Absorber

Virtual Sound Sources Inputs 5

Enable ■ Boid Number 5
Inertia 4. Attr. Neighbors 5
MaxSpeed 2. MinSpeed 1.
Centering 1. Acceleration 0.01
Matching 0.25 Avoid 3.
wall Avoid 2. Edge Dist. 0.01
Speed 5. Neigh. Dist. 3.5

Figure 5: a screen capture showing a Max patch using the Boids module with the IVES Room, and Soundfield module to steer sound sources in a virtual environment.

Use Cases

The deep integration of these two modules into the IVES engine, in addition to the other possibilities offered by IVES, allows multiple applications and opportunities for autonomous and improvisational motion within audiovisual virtual works. Improvised performances, conducted compositions, and also virtual installations can be realized. The behavior of the autonomous agents can be related to a first-person user, such as a performer or visitor, other agents, and/or virtual objects to generate improvisational trajectories or simulate lifelike behavior in virtual worlds as an artistic work.
Possible application scenarios can, but are not limited to, the following types and combinations:

User/listener-dependent behavior:

Video 1: a screen recording of an autonomous agent using the Agent module steering a virtual object and sound source. The behaviors seek, flee and arrive are automated over time in regard to the first-person user's position. (Use headphones for binaural audio)

With the Agent module, the behavior of an autonomous agent can be used to control a virtual object and/or sound source to create dynamic and improvisational trajectories with a first-person user as the target. By automating the steering type and its properties, the behavior can be further controlled for use in a live performance (see Vid.1).

Complex combinations of behavior:

The combination of multiple agents with different behavior towards each other and/or their environment can be used to realize complex scenarios for the motion of virtual objects and/or sound sources.

Video 2: a screen recording showing a ‘loving birds’ example. With one virtual object wandering around a virtual room, seeked by another object. Both are steered by autonomous agents using the Agent module.

Linking individual agents to each other and to their environment enables life-like improvisational behavior for artificial life simulations. In this example of ‘loving birds’, one agent wanders around the spatial boundaries defined by the IVES Room module, while a second agent floats around the first agent, which is the target of its seek behavior.

Video 3: a screen recording showing two autonomous agents using the Agent module. One is path following a 3D path, while being pursued by another one. Both are steering a virtual object as well as sound source. (Use headphones for binaural audio)

Path following in combination with other behaviors can be used to realize complex trajectories with improvisational deviations of multiple virtual objects and/or sound sources. In this example, the IVES Path module was used to create a path that is followed by one agent in the path-following behavior, while a second agent pursues the first one.

Group behavior of multiple agents:

Multiple agents can be combined to perform a group behavior in the environment. With the flocking behavior of the Boids module, the behavior of multiple agents can be used to control virtual objects and/or sound sources in their environment. Additional agents using the Agent module can be attached to extend the flocking behavior (see Vid. 4):

Video 4: a screen recording of a swarm of 5 sound sources in flocking behavior using the Boids module, being pursued by a virtual object in seek behavior using the Agent module. (Use headphones for binaural audio)

Discussion

Steering behaviors of autonomous agents are used in many ways in composition practice. From the generation of complex trajectories [8] to the simulation of swarm behavior [26], there are a variety of approaches. As Graham Wakefield has pointed out, there is particular potential for audiovisual virtual works in the context of Artificial Life simulation and computational world-making [27]. Graham's Cosm toolkit [28] already provided a 3D engine in Max integrating the simulation of autonomous agents. The already long-outdated Cosm toolkit left a gap not only for a 3D engine in Max, but also for the integration of autonomous agents into such an engine.

The modules for autonomous agents presented in this paper, especially in conjunction with the IVES 3D engine, not only fill a part of this gap, but also provide new approaches to interacting with autonomous agents for composition and performance practice. The possibilities of not only isolated behavior of agents towards each other and their environment, but especially in interaction with a first-person user in virtual environments, offer many possibilities for compositions, performances, or virtual installations (see Use Cases). In contrast to such an implementation within a common game engine, the integration into an established environment for audiovisual digital art like Max offers special advantages. Artists can combine the design of virtual environments with autonomous agents, for stage performances or VR, with already known processes such as generative design for sounds and virtual objects in a familiar environment tailored to art production. In addition to the potential that lies in the creative use of the capabilities of both presented modules: behaviors can be combined in different ways and applied to different targets; users can also customize the implemented behaviors or integrate their own developments due to the open-source architecture of the modules. In their current state, the autonomous agents are limited to basic steering behaviors and thus to simple movements. While this allows for the dynamic and improvisational integration of multiple objects and sound sources for audiovisual compositions and performances, it quickly reaches its limits for the realization of complex behaviors and artificial life simulations with large numbers of agents. In order to extend the possibilities with autonomous agents in IVES, the integration of additional and new steering behaviors is considered in the further development of the presented modules. In the next step, the steering behavior of the autonomous agents will be combined with evolutionary algorithms in order to realize even more complex Artificial Life simulations. In addition to the modules for autonomous agents, also the development of further AI modules for spatial composition and performances in IVES is planned for the future. The presented modules are available as part of the IVES toolkit in the official Github repository: https://github.com/AudioGroupCologne/IVES

Acknowledgments

I thank Christoph Pörschmann (TH Köln) for supervision and support in carrying out the research and writing this paper.

Ethics Statement

Since the main code of this work is based on the open-source code of Craig Reynolds and Daniel Shiffman, not only appropriate credits are given, but its implementation, as well as the whole IVES project itself, is open-source software without any profit intentions or restrictions. It is a continuation of the work of the authors mentioned above, and thus not only a further contribution to the open-source community, but also of benefit to the art and scientific community.

References

- B., Nadir. (n.d.). *Spat Trajectories*. Retrieved 10.03.2023, from https://github.com/nadirB/Trajectory_Score_Library ↩
- Baalman, M. A. J. (2010). Spatial composition techniques and sound spatialisation technologies. *Organised Sound*, *15*(3), 209–218. https://doi.org/10.1017/S1355771810000245 ↩
- Blackwell, T. M., & Bentley, P. (2002). Improvised music with swarms. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, *2*(February 2002), 1462–1467. https://doi.org/10.1109/CEC.2002.1004458 ↩
- Carpentier, T. (2018). A new implementation of Spat in Max. *Proceedings of the 15th Sound and Music Computing Conference: Sonic Crossings, SMC 2018*, 184–191. ↩
- Carpentier, T., & Gerzso, A. (2021). *Steering Behaviors for Spatial Sound Authoring*. *Proceedings of International Computer Music Conference*. ↩
- Coduys, T., & Ferry, G. (2004). Iannix - Aesthetical/Symbolic visualisations for hypermedia composition. *Sound and Music Computing, 33,* 1–6. ↩

- Cycling’74. (n.d.). *Max*. Retrieved 10.03.2023, from https://cycling74.com/ ↩
- Dziwis, D., Arend, J. M., Lübeck, T., & Pörschmann, C. (2021). IVES - INTERACTIVE VIRTUAL ENVIRONMENT SYSTEM: A MODULAR TOOLKIT for 3D AUDIOVISUAL COMPOSITION in MAX. *Proceedings of the Sound and Music Computing Conferences*, *2021-June*(4), 330–337. ↩
- Dziwis, D., Lübeck, T., & Pörschmann, C. (2023). Modular Room Simulation for the IVES 3D Engine. *Forum Acusticum - 10th Convention of the European Acoustics Association (Accepted for Publication)*, 1–8. ↩
- Epic. (n.d.). *Unreal Engine*. Retrieved 10.03.2023, from https://www.unrealengine.com ↩
- Franklin, S. (1997). Autonomous Agents as Embodied AI. *Cybernetics and Systems*, *28*(6), 499–520. ↩
- Franklin, S., & Graesser, A. (1996). Is It an Agent, or Just a Program?: A Taxomony of Autonomous Agents. *International Workshop on Agent Theories, Architectures, and Languages*, 21–35. ↩
- GEMEM. (n.d.). *Holophon*. Retrieved 10.03.2023, from http://dvlpt.gmem.free.fr/web/static.php?page=Holophon_main ↩
- Miyama, C., Dipper, G., & Brümmer, L. (2016). Zirkonium 3.1 - A toolkit for spatial composition and performance. *ICMC 2016 - 42nd International Computer Music Conference, Proceedings*, 312–316. ↩
- Møller, H. (1992). Fundamentals of binaural technology. *Applied Acoustics*, *36*(3–4), 171–218. https://doi.org/10.1016/0003-682X(92)90046-U ↩
- Pattie, M. (1995). Artificial Life Meets Entertainment: Lifelike Autonomous Agents. *Communications of the ACM*, *38*(11), 108–114. https://doi.org/10.1145/219717.219808 ↩
- Processing Foundation. (n.d.). *p5js*. Retrieved 10.03.2023, from https://p5js.org/ ↩
- Processing Foundation. (n.d.). *Processing*. Retrieved 10.03.2023, from https://processing.org/ ↩
- Reynolds, C. W. (1987). Flocks, Herds, and Schools: a Distributed Behavioral Model. *Computer Graphics (ACM)*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406 ↩
- Reynolds, C. W. (1999). Steering behaviors for autonomous characters. *Game Developers Conference*, 763–782. ↩
- Shiffman, D. (n.d.). *Stay within walls example*. Retrieved 10.03.2023, from https://github.com/nature-of-code/noc-examples-processing/blob/master/chp06_agents/NOC_6_03_StayWithinWalls/Vehicle.pde ↩
- Shiffman, D. (n.d.). *The Nature of Code 2*. Retrieved 10.03.2023, from https://thecodingtrain.com/tracks/the-nature-of-code-2 ↩
- Shiffman, D., Fry, S., & Marsh, Z. (2012). *The Nature of Code*. D. Shiffman. ↩
- TripinLab. (n.d.). *Sound Trajectory 2*. Retrieved 10.03.2023, from https://www.tripinlab.com/ ↩
- Unity. (n.d.). *Unity Engine*. Retrieved 10.03.2023, from https://unity.com/ ↩
- Valentino Braitenberg. (1986). *Vehicles, Experiments in Synthetic Psychology* (p. 168). The MIT Press. ↩
- Wakefield, G., & Ji, H. (2009). Artificial nature: Immersive world making. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5484 LNCS*(October 2017), 597–602. https://doi.org/10.1007/978-3-642-01129-0_68 ↩

- Wakefield, G., & Smith, W. (2011). Cosm : a Toolkit for Composing Immersive Audio-Visual Worlds of Agency and Autonomy. *Proceedings of the International Computer Music Conference, 2011*. ↩