# The In-Game AUV-Model and its Visualizations in Travis

Ingmar Ludwig
*University of Bremen*
Bremen, Germany
iludwig@uni-bremen.de

Felix Haffke
*University of Bremen*
Bremen, Germany
fhaffke@hfk-bremen.de

Lennart Schunk
*University of Bremen*
Bremen, Germany
lennart_schunk@web.de

***Abstract*—This paper describes the AUV-model and the visualizations for its status used in the Travis simulation. The first part describes how the AUV-model is build in regards to its meshes, textures and the implementation of its propeller- and hydroplane movements. The second part describes how the AUV moves through the simulation and what needs to be considered while dealing with the movement method. The final part gives an overview over the different visualizations of the AUVs status, describes what the underlying ideas are and how the implementation is structured.**

***Index Terms*—unreal engine, unreal, trajectory, travis, visualisation, AUV-model**

## I. Introduction

Purpose of the Travis simulation is to replay and visualize real dives of the MARUM-SEAL AUV. The documentation is split in several papers. In this paper the AUV-model, its components and the visualizations of its status are described.

This paper is divided in three parts. In the first part the origin of the different components of the AUV-model are described (e.g. what is the source of the data is and how the components were created). In the second part the implementation of the movement and rotation of the AUV-model and what needs to be considered while dealing with the implementation is illustrated. In the final part the visualizations of the AUVs status (e.g. pressure or motor-RPM) are explained relating to their fundamentally ideas and implementation structure.

## II. AUV-Components

The AUV-model consist of several distinct parts which are discussed in regards to their origin, basic idea and implementation.

### A. Mesh

To create a realistic and lifelike model of the AUV, pictures of the actual AUV taken from the offical MARUM website [1] were used. From these a simplistic model was made in blender. Based on a tube shaped body, a simple tower and hydroplanes more and more details were added to the AUV model iteratively. Those details include individual screws, antennas, lights and more.

To ensure the movability according to the data the AUV was separated into several parts (body, wings, tower and details).

[1]https://www.marum.de/Entdecken/AUV-MARUM-SEAL.html

### B. Texturing

In the simulation the model is covered with textures. The texture was created without the use of real images. This leads to a more unrealistic texture but ensured a much more clean and polished look which again leads to a more user friendly and easier to understand visualisation. To create the texture a simple base color was used and then combined with some details and a texture that we took from texturelib.com[2]. The logos where added and ambient occlusion was baked directly on the texture to get rid of performance issues which may occur by using an ambient occlusion effect. To make the AUV look more used and weathered different layers of dirt and irregular surface were added to the texture. In addition to that the logos were made looking more run down and a basic metal texture was added.

The material work was done in the unreal engine using base texture, ambient occlusion texture and customized variables for shininess, metallic and roughness. The materials where made to represent the real metal of the AUV as accurate as possible.

The resulting AUV-model (see figure 1 and 2) seeks a good compromise between a realistic and believable look and the visualisation purpose.

### C. wing and propeller movement

To enable realistic movement of the hydroplanes and the tower, these elements were modelled separately in blender and than later animated in the Unreal Engine using blueprints. Goal of the animation was to move the hydroplanes and the propeller as close as possible as at the real AUV. To enable that each of the items was placed at the AUV using photos of the AUV from the MARUM Website [3]. After that the models were transformed into blueprints, setting each part as a child actor of the main AUV body.

Finally each piece got its own animation function. All Functions are then called in the level blueprint using custom events for inter-blueprint communication.

### D. AUV size

The size of the AUV in Travis has to take into account several demands that are contradictory. To best satisfy those

[2]http://texturelib.com/texture/?path=/Textures/metal/bare/metal_bare_0029
[3]https://www.marum.de/Entdecken/AUV-MARUM-SEAL.html

demands while still keeping the simulation clear, three different sizes for different demands were created that are selectable through an interface button. The smallest size (see figure 3) represents the approximately correct size of the AUV in regards to the map an thus makes the true size ratio visible. Since this size is very small and thus makes it difficult to see the visualizations, a second, bigger size was added (see figure 4). A third size was added to answer the demand for a good possibility for seeing the movement of the AUV from a great distance that enables to see the whole map (see figure 5). However this very large size makes it very difficult to see the connection between the trajectory and the AUVs movements, so the medium size remains necessary.

Due to wishes from the MARUM a second size-related functionality was added: while the keyboard button g is pressed the AUV is temporally increased ten times in size. This enables the user to easily find the AUV if he or she temporally lost the view on it.

Since the size of the earth surface represented by the map differs from mission to mission, the smallest AUV size has to be adjusted accordingly (for further information over the correlation between the earth surface size and the map size refer to the Aquarium-paper). Initially the AUV has the size that would be right if one Unreal Unit would represent one meter. This size is than adjusted by multiplying the number of Unreal Units that represent one meter in this special setting (which is passed from DataCommunicator).

A known issue with the AUV sizing in Travis is that the hitbox of the AUV-model does not resize correctly. This leads to limitations in getting really close to the smallest AUV-model version and enables the user to fly through the largest version. Since this fact is no real problem while using the simulation, it was not fixed due to other priorities.

## III. AUV Movement and Rotation

The purpose of the implemented AUV movement functionality is to be able to replay AUV dives with continuous and frequent data points (at least one per second, tested with 10 data points per second). General idea is to take those data points and linearly interpolate the movement and rotation between them.

All the movements and rotations of the AUV-model take place in the Move AUV method, which takes the position and rotation data from two data points and an interpolation value. The method than interpolates the position and rotation values and places the AUV-model accordingly in regards to its center of gravity. The AUV-models center of gravity on the other hand was placed directly in the geometric center of the model during the creation process.

The method excepts two kinds of z coordinates, one being the actual z component and the other being the altitude above ground. In the current simulation only the first one is used, but since it was not clear what the data would look like at the beginning of the project when the method was implemented, both possibilities were anticipated.

A general problem with the approach of using no intelligent steering for the AUV, but instead setting the positions directly according to the data from the log file, is that effects from the data logging phase in the real dive can lead to strange effects in the simulation. In the data samples we got for testing our simulation the longitude, latitude and the depth seem to have different update rates, leading to a slight zigzag curse when the simulation is played with low speed (This zigzag does not result from the interpolation process: It also happens when interpolation is turned of. In this case the AUV is teleported from one point to the other in a zigzag fashion.). But since our task was to visualize the *data* (and not necessarily the real dive), we chose this kind of AUV positioning nevertheless.

## IV. Changing the AUV Model

For changing the AUV-model (a feature that is planned for the future), several steps are necessary: First, the new AUV-model needs to be transferred into an unreal uasset file using the unreal engine. Than the current AUV-model (named ”submarine” in the simulation) can be switched to the new model. If there is a wish to use the motor rpm, pressure, speed or rotation visualization, the actors for the visualization (which also have these names and are structured accordingly) need to be moved from the old to the new model. If the hydroplane rotation also should be visualized like in the current model, these actors also need to be transferred to the new model and, since the position of the hydroplane will probably differ, repositioned. If the hydroplanes also should move like in the current model, the new model will need to use the Move Front Left Plane, Move Front Right Plane, Move Back Left Plane, Move Back Right Plane and Move Submarine Tower Events for moving its hydroplanes. Since we were not able to check this procedure (due to the lack of another AUV-model), this guidelines should be treated with caution.

## V. Visualization of AUV-Status

In addition to displaying mission data as text in the user interface, we wanted to visualise important parts of the data within the 3D space. To archive a clean and easy to understand visualisation it was placed directly around the auv. The different visualizations and their general ideas are described in this paper.

### *A. AUV rotation*

We started with the AUVs rotation within the 3D space. First we thought about what we had to display: 3 Rotation axes and a way to show at what degree the AUV is rotated within those axes. The first thought was to create some kind of 3D compass and use the AUV itself as the compass needle. It was planned to archive that by creating a model of the 3 axes and then move those with the AUV. The compass would only move with the AUVs position but not with its rotation. This would result in the AUV rotating within the compass. This idea whatsoever did not quite work out due to the fact that the AUV is usually not pointing at the axis. As we figured out, we had to rotate the axes with other axes in the right way to ensure a correct display of the data. For example the pitch is rotated with the heading angle to ensure that the AUV is always pointing at the correct pitch axis (while still being able to rotate horizontally). In the end we still somehow used the 3D compass idea: The AUV is within a "sphere" build out of the 3 rotation axes (xyz) for pitch, yaw and heading angle (see figure 6, 7 and 8). Those move with the AUV and an arrow connected to the AUV points at the rotation scales.

### *B. Pressure*

Next the pressure visualisation was created (see figure 9). The goal was to keep the visualization simple and easy to understand. We added a sphere around the AUV which changes its scale and color as the pressure changes. Low pressure results in a big blue sphere and high pressure results in a small red sphere which seems "pressed together". For better discriminability later a progressive color flow was added going from blue over green to red using the whole RGB spectrum.

### *C. Speed*

For visualising the speed we added an arrow right in front of the AUV (see figure 11). The arrow rotates and moves with the AUV but changes its size according to the AUVs speed: As the AUV gets faster, the size of the arrow increases, as it looses speed the arrow decreases in size.

### *D. Propeller speed*

The propeller speed visualisation is almost the same as the speed visualisation but instead of being in front of the AUV, the arrow is placed behind it (see figure 11). As the propeller speed increases the arrow gets bigger. The AUVs movement speed is obviously related to the propeller speed and the design of the visualizations tries to echo that by making the interaction visible. Also using both visualizations together enables the user to easily recognise some malfunctions: For example if there is a big arrow behind the AUV and a small arrow in front of it, it is obvious that the AUV is not building up speed while the propeller is moving fast. This might be due to a damaged propeller.

### *E. Hydroplanes rotation*

The Hydroplane rotation is displayed similar to the AUVs rotation itself. But because the Hydroplanes only have one degree of freedom each, instead of using a compass with 3 axes only one axis was used. For visualizing the rotation an arrow is moved together with the hydroplane. Together with a (in reference to the AUV) static scale, the degree of the rotation is visible (see figure 10).

### *F. implementation*

All visualizations share a similar structure to keep the Level Blueprint as clean as possible. All visualization are based on objects in the Level Blueprint which are updated within every tick (if the visualizations are visible). Each of those updates is connected to the Tick Event in the Event Graph of the Level Blueprint and highlighted as a interrelated segment with a comment box.

The updates are done using Unreals Set Actor Scale 3D, Set Actor Relave Location and Set Actor Rotation methods after the right scale, rotation and location was calculated using the current Waypoint Data from the DataCommunicator.

The visualizations are activated using buttons in the interface Blueprint. This is realized utilizing a cascade of Events that seems necessary due to Unreals structure. When a button is clicked in the interface, a On Clicked Event is called automatically. This Event is than connected to an Event Dispatcher of the origin Blueprint that enables calling events in different Blueprints. But before this is possible, a Custom Event of the destination-Blueprint, in this case the Level Blueprint, has to be bound to the Event dispatcher. This happens in the Event Begin Play section of the Event Graph in the Level Blueprint. In this Custom Event in the Level Blueprint the main functionality for showing or hiding the visualizations is located (which is located in the initialization block of the Event Graph section).

### *G. Placement of the visualizations in unreal*

There are two natural places for the visualizations in unreal: Within the level and within the AUV-model blueprint. Both of these places have their advantages. While placing the visualizations within the AUV-model would lead to cleaner structure, a placement inside the Level Blueprint makes it easier to change the AUV-model in the future. This is due to the fact that most of the visualizations can stay the same with every AUV model while only the visualization of the hydroplane rotations need repositioning (which is easily achievable). This way redundancy in the Blueprints while using multiple AUV-models can be avoided. Since multiple AUV-models are an important goal for the future, the second variety was chosen.

## VI. Appendix

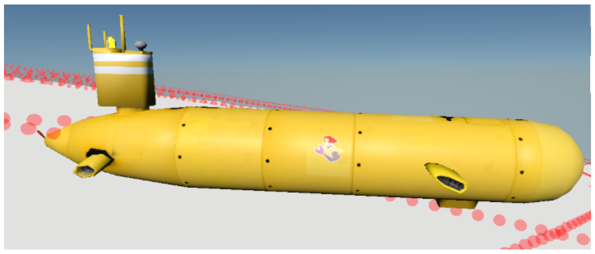

Fig. 1. The AUV-model as seen from the side

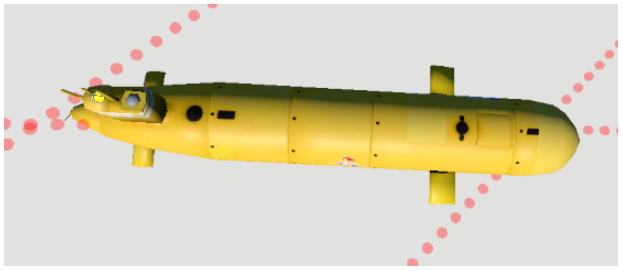

Fig. 2. The AUV-model as seen from the top

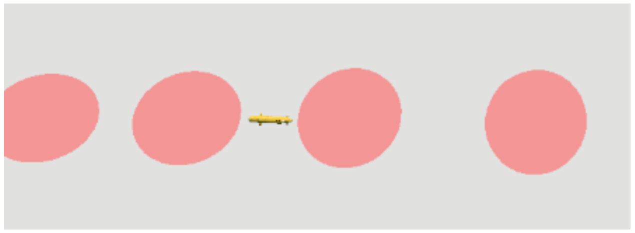

Fig. 3. The AUV-model in the smallest, in regards to the map correct size

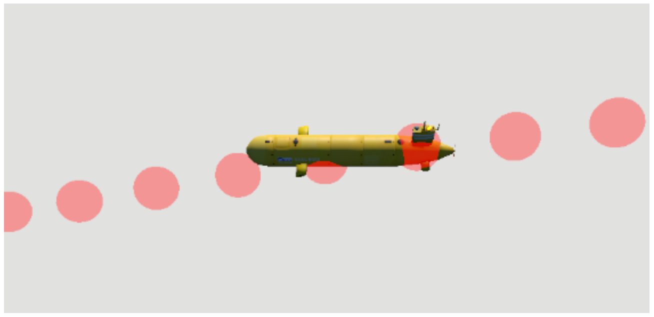

Fig. 4. The AUV-model in the medium size

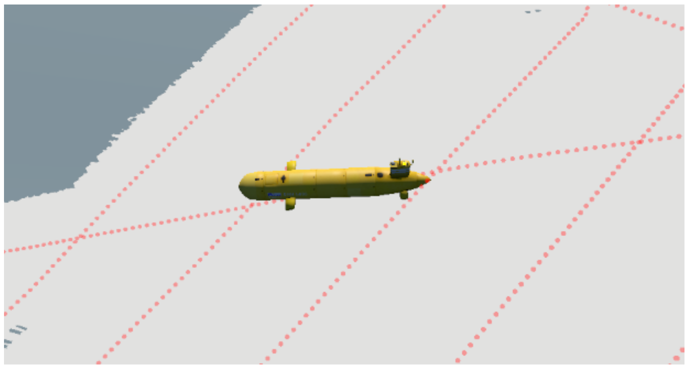

Fig. 5. The AUV-model in the largest size

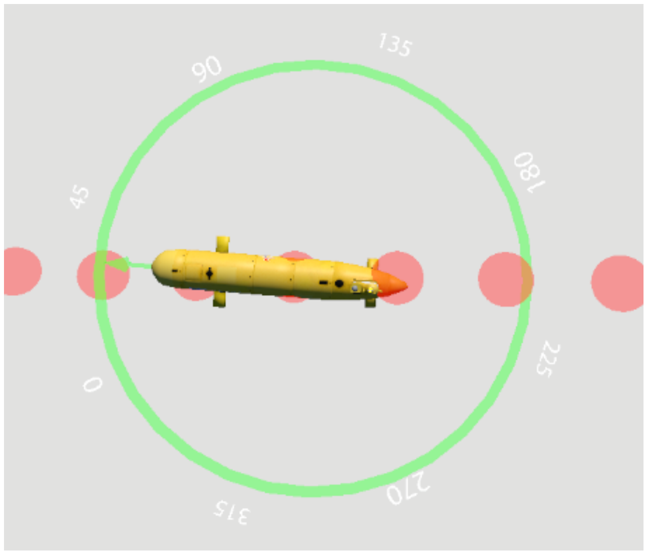

Fig. 6. The visualization of the heading angle (movement direction)

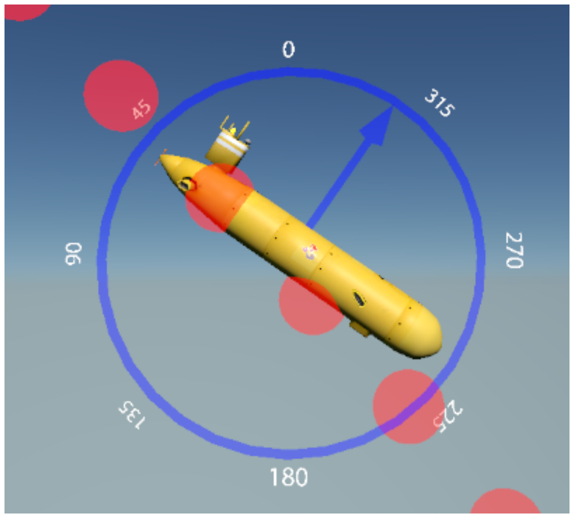

Fig. 7. The visualization of the AUVs pitch angle

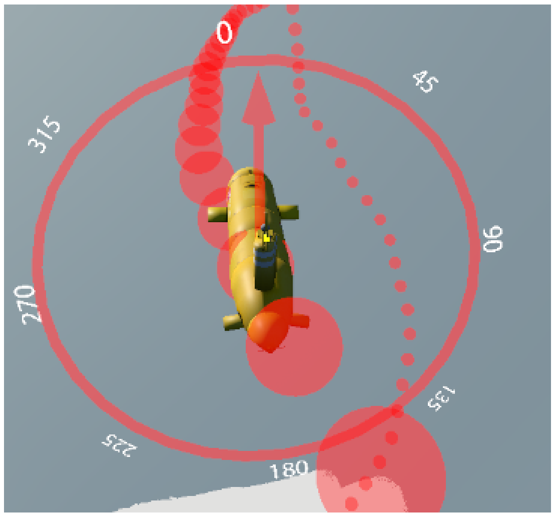

Fig. 8. The visualization of the AUVs roll angle

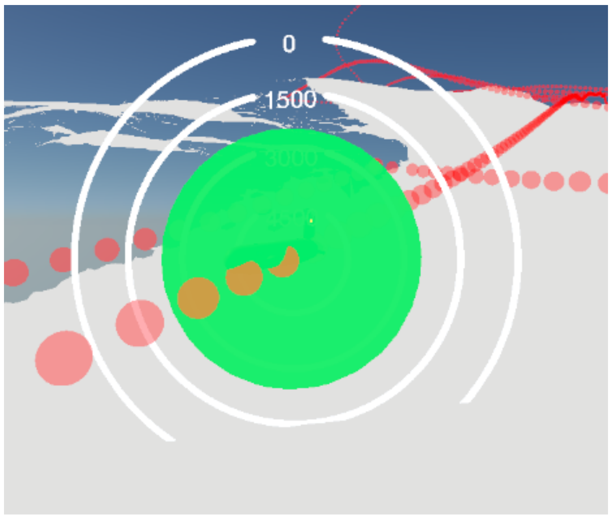

Fig. 9. The visualization of the pressure on the AUV

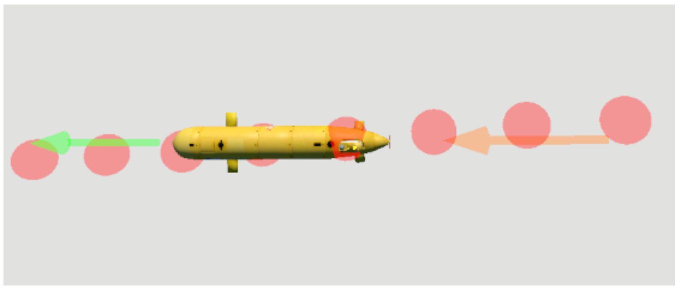

Fig. 11. The visualization of the speed (green arrow) and rotations per minute of the motor (orange arrow)

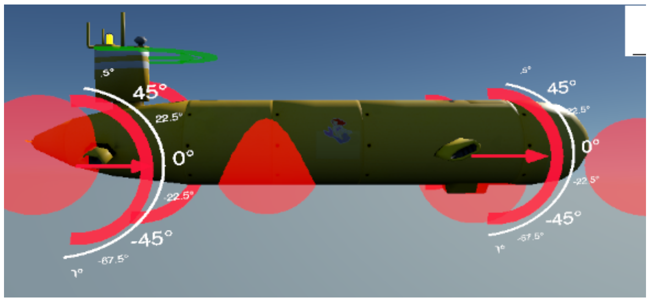

Fig. 10. The visualization of the AUVs hydroplanes rotation status