Dynamic Split-Screen for Visualizing Events in Augmented Reality

Luca Becheanu (L.Becheanu@student.tudelft.nl) | Supervisors: Elmar Eisemann, Mark Winter

INTRODUCTION

Sailing+ is an AR/VR Application for experiencing **sailing regattas.** It requires a **captivating way** to visualize competitors and key moments of **an event**.

The user has **complete** control of the camera as he envisions the race.

"How can a virtual camera system engagingly exhibit interesting race events, in an interactive AR/VR setting?"

Two primary points of focus (user/event) need to be handled at the same time. Optimizing the screen space is done using Voronoi Diagrams (Figure 1).

Dynamic split-screens use a **line separator** (Figure 2.ab) that **disappears** when the points get close to each other (Figure 2.c).

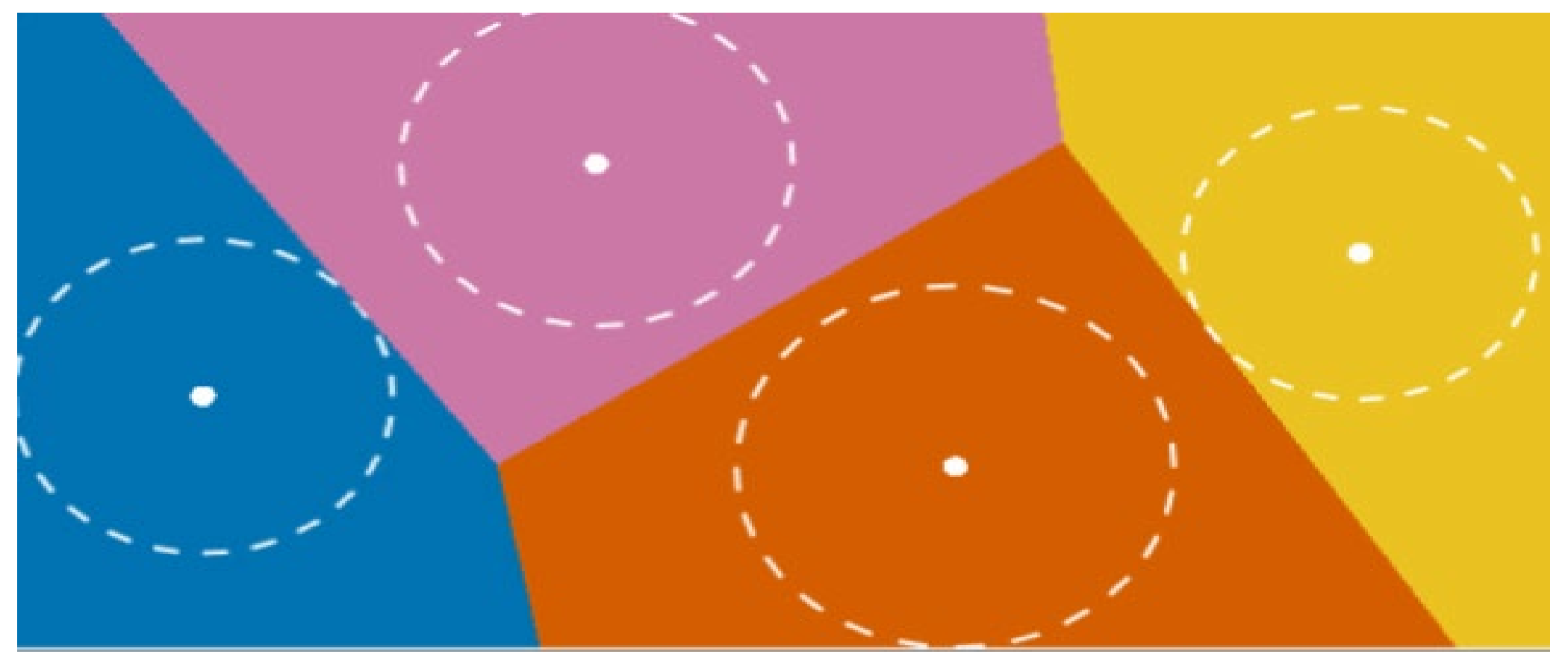
Figure 1. Voronoi diagram with multiple Voronoi cells, which consist of every point in the Euclidean plane whose distance is less than or equal to its distance to any other cell center.

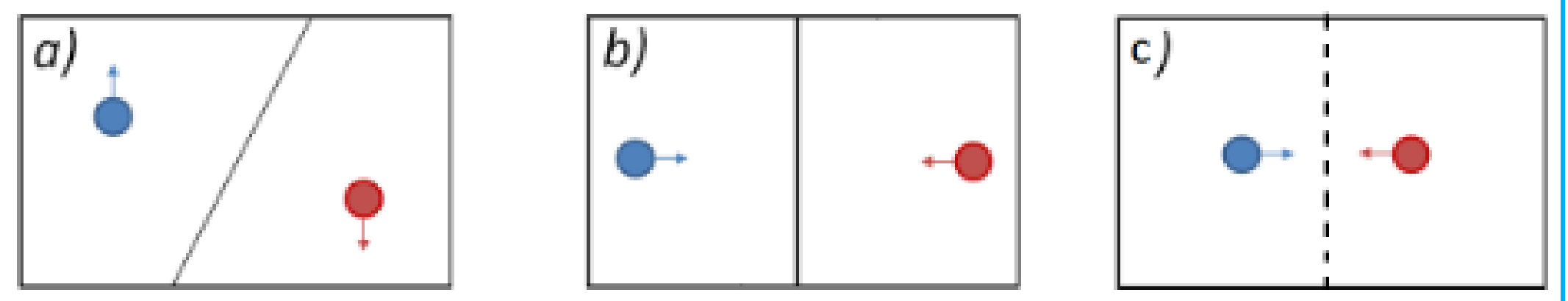

Figure 2. (a) The screen is split to show both points (red and blue) when they are far away from one another. (b) The separator line changes its orientation dynamically based on the points' positions in world space while they move (arrows). (c) The separator line vanishes once the points move towards each other and are close by.

METHODOLOGY

SPLIT PHASE

If the 3D distance between cameras is higher than a **maximum** edge of an interval defined in the algorithm, the split phase occurs (Figure 6).

We **normalize** the world camera positions into screen space such that the center of the Voronoi cells (M, E in Figure 3) are always on opposite sides. Then, we compute which pixel is **closest** to what camera.

The cell centers (M, E) follow the changes in the positions of the cameras in the 2D screen space and the split line separator always changes based on where a camera is in the plane **relative** to the other.

The event is always shown in the middle of its cell, by **deviating** all pixels that belong to the event cell.

The user is guided towards the event by using an **arrow** that points towards the position of the event.

A **slider** also appears as an engaging feature that can be moved in the direction of the camera that the user would rather see (Figure 8).

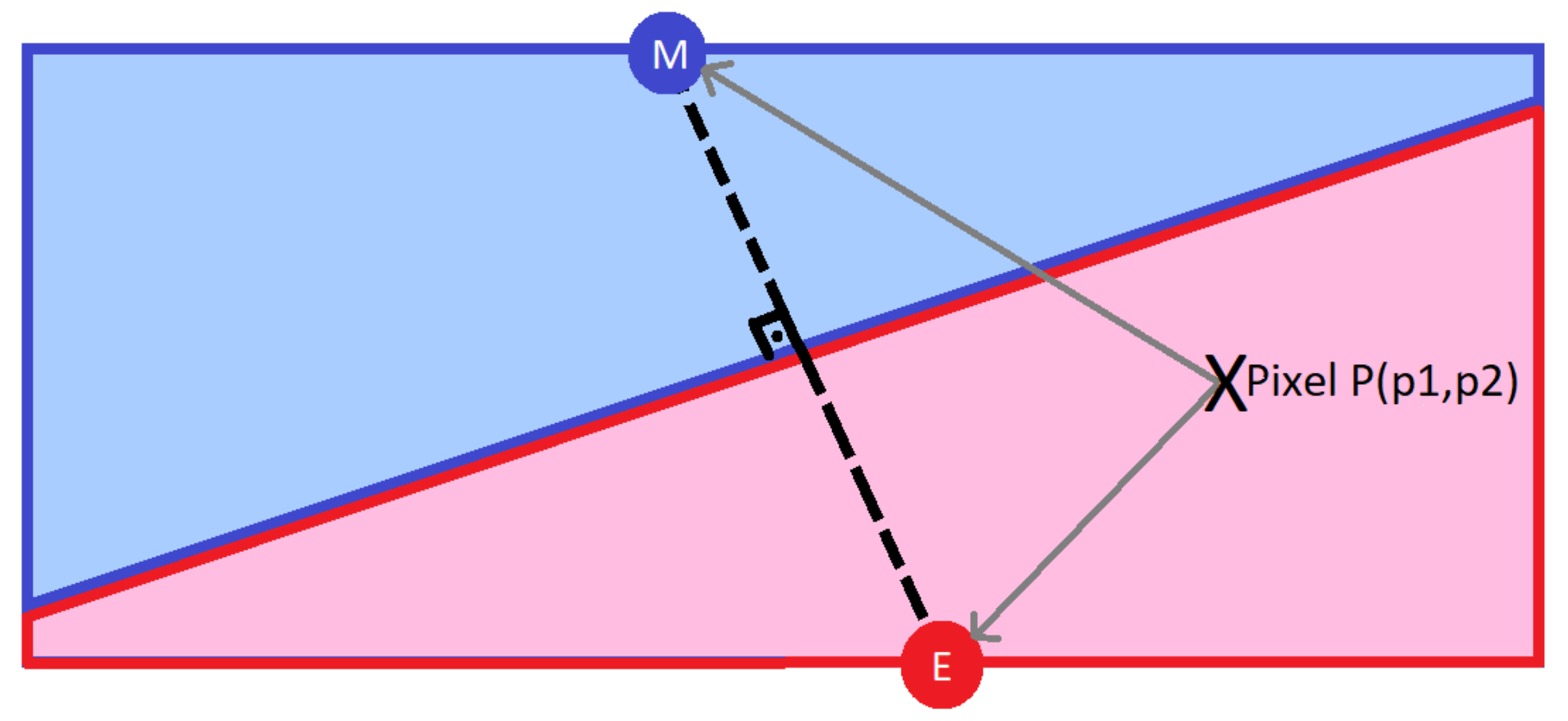

Figure 3: Pixel P belongs to the Voronoi cell with center in E (Event camera) because it is at a smaller Euclidean distance than the Voronoi cell with center in M (Main camera). The pixels that are on the blue/red line between the two cells are at the same distance between the center of these cells and form the separator line.

TRANSITION PHASE

The transition is the intermediate phase between the split phase and the merge phase and consists of **raising the screen space** allocated to the main user's camera detrimental to the event camera (Figure 7) using the Weighted Generalization of the Voronoi Diagram.

Using a Hermite Interpolation we increase and shrink the event cell along with the separator line as **smoothly** as possible.

The main cell will be translated in M' (Figure 4), to be able to attract like a magnet more pixels to the user camera (Figure 7a).

The separator line between cells is created by translating M' to M'' and will shorten depending on the interpolation (Figure 7b).

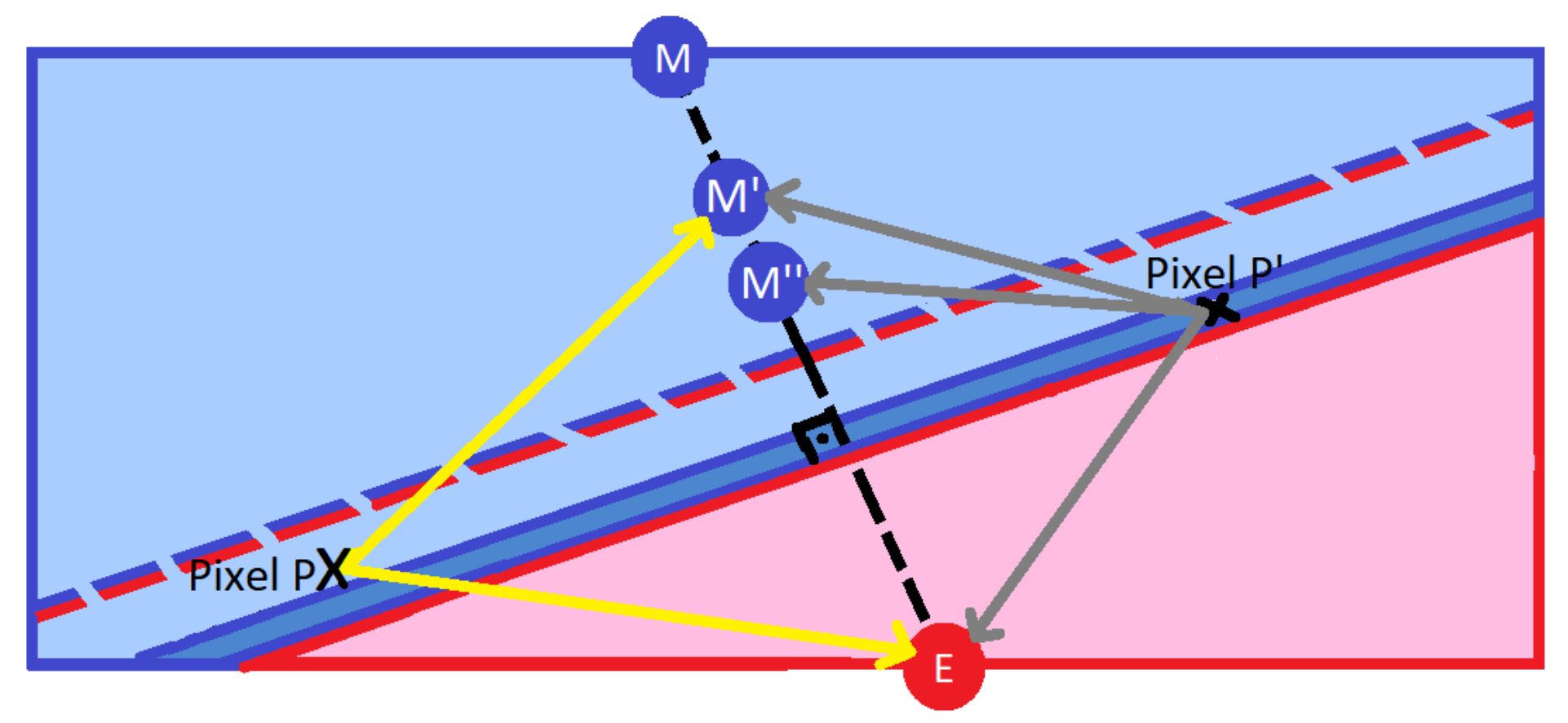

Figure 4: The cell center M (Main camera) is translated in the point M' (new cell center of the Main camera for the transition phase) and M'' (used for computing the line) respectively, in the direction of the event camera. Pixel P' is closer to M', rather than E (Event camera), therefore it belongs to the Main cell. Pixel P' is closer to E than M', therefore it belongs to the Event cell. Because it is closer to M'' than E, it is colored in dark blue to form the separator line between cells.

MERGE PHASE

The merge phase happens when the distance is less than the **minimum** value of the defined interval. The user can still choose to look in a different direction than where the event is taking place. Due to this fact, we need to take into account multiple key points (Figure 5):

1. The 3D distance between the point where the forward vector of the main camera intersects the regatta field and the competitor that the event camera is pointing towards (blue and red lines). If it's greater than a maximum visible distance, a new split phase occurs.
2. The angle between two 3D vectors, the main camera to the center of the viewport and the main camera to the point where the event competitor in the regatta field is (angle α). If it's greater than a maximum visible angle, a new split phase occurs.
3. The 2D normalized distance between the main camera center of the viewport and the event competitor (purple).

To handle the new split and transit phases that are formed, we simulate the movement of the camera positions along the edge of the screen (Figure 4) by **normalizing** the coordinates if they are outside of the viewport of the main camera.

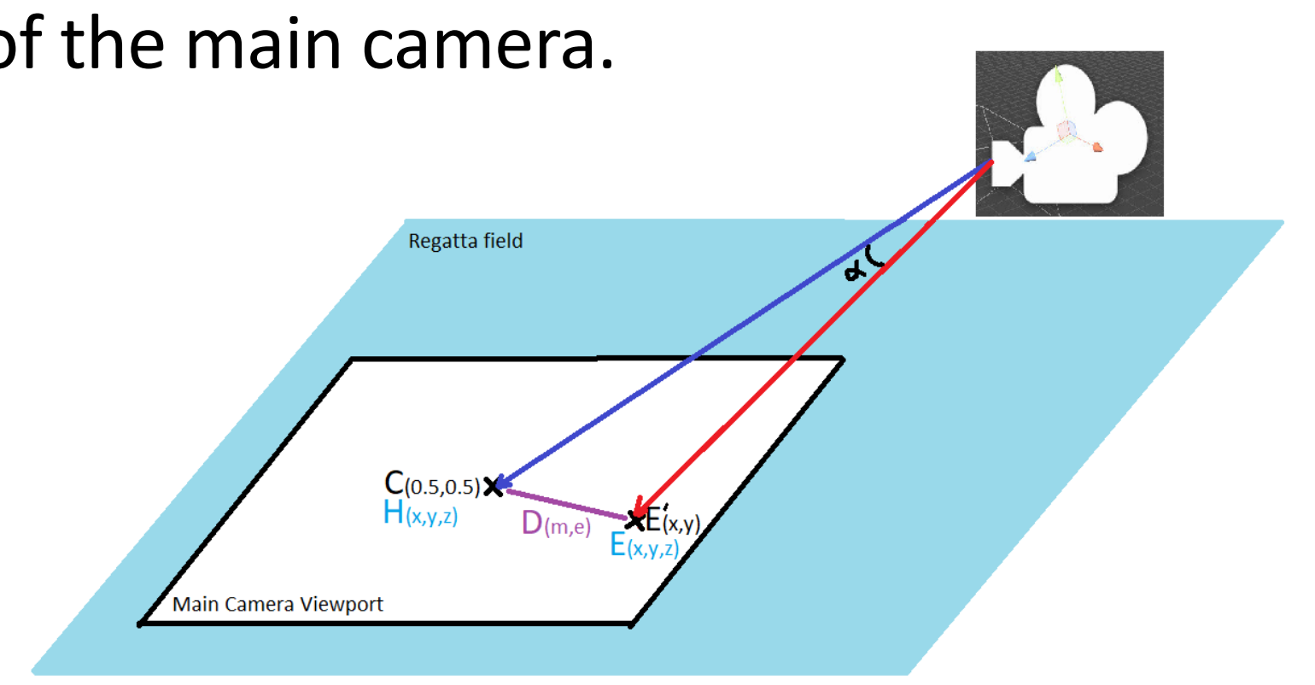

Figure 5: Main camera pointing at the regatta field. H is the point with world coordinates where the forward vector (dark blue) going from the camera intersects the regatta field. E is the point with the world coordinates of the event competitor. C is the center of the viewport (white) with coordinates (0.5, 0.5), and E' is the point with the coordinates of the event competitor.

DISCUSSION

Our algorithm performs in O(m · n) time complexity where m is the number of active cameras (Voronoi cells) and n is the number of pixels. It is a negligible issue for two cameras and it will not affect the framerate, making the algorithm fast enough for the required seconds in which the event is taking place.

The technique allows the customization of controlling the split-screen via several parameters. Firstly, the edges of the distance intervals can be modified to better delimit when the event is still visible enough to be visualized and when the transition leading to the full split should begin and end. The colors of the separator line and the arrow pointing toward the event as well as the line thickness can also be changed.

The solution presents a split for two cameras, but it can be extended for multiple events happening at the same time by adding more event camera cells. The difficulty is in creating a fair size balance when more points of focus come into play (Figure 1), and an even ratio solution for more than two points has been proven to not exist.

REFERENCES

Voronoi Diagrams: Marjorie Senechal, Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. 1995. Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. *The College Mathematics Journal 26*, 1 (Jan. 1995), 79.

Weighted Generalization of the Voronoi Diagram: Peter F. Ash and Ethan D. Bolker. 1986. Generalized Dirichlet tessellations. *Geometriae Dedicata* 20, 2 (April 1986), 209–243.

Hermite Interpolation: Natalya Tatarchuk. 2003. Advanced Real-Time Shader Techniques. AMD

Figure 1: Squirrel Eiserloh. 2016. Math for Game Programmers: Juicing Your Cameras with Math. In *Game Developers Conference*.

Result Figures: Original images from the Sailing+ Application.

RESULTS

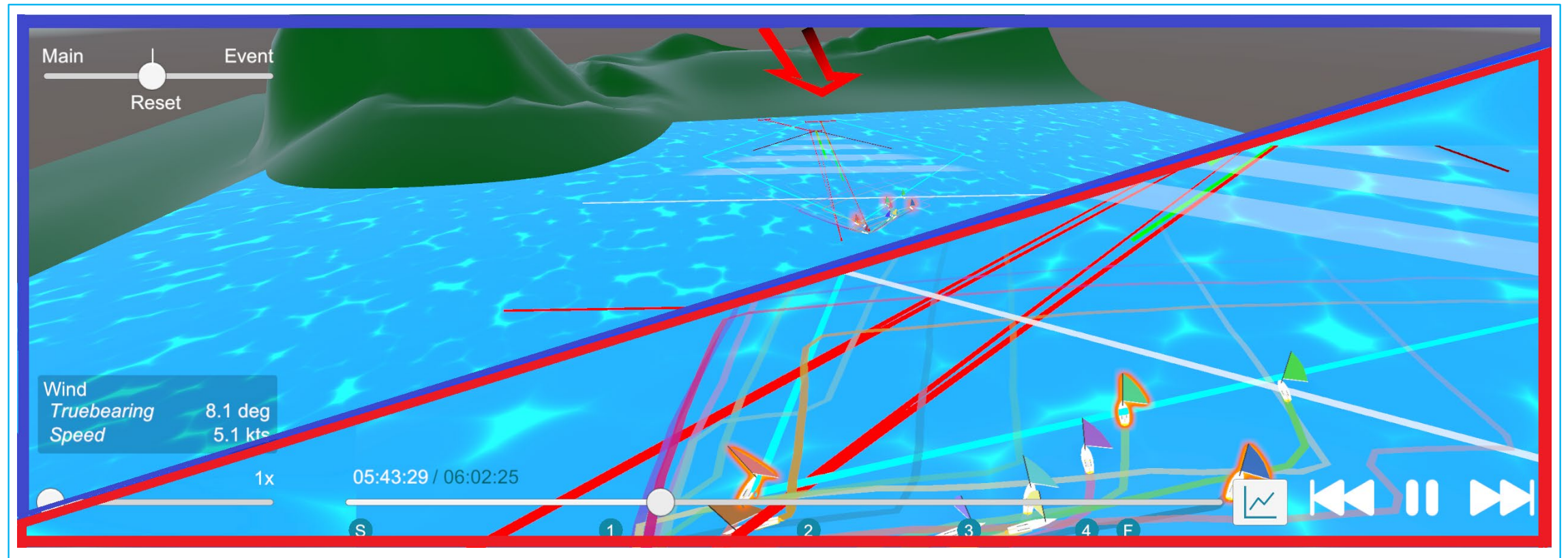
Figure 6: A dynamic separator line (middle) changes its orientation based on the position of the user's camera (highlighted with blue, left), which is far away from the race, relative to the event camera (highlighted with red, right) in order to have an equal screen distribution.

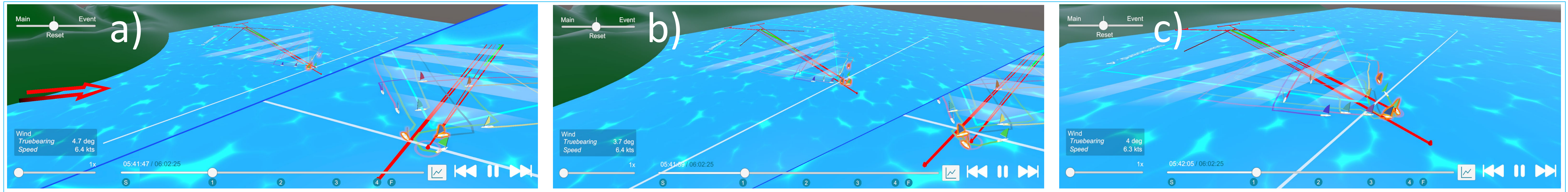

Figure 7: Magnetism effect of the main camera, which attracts more pixels to it as the distance between itself and the event camera decreases. (a) Main camera view (left) increases as the distance between itself and the event camera (right) decreases. (b) The separator line as well as the event (bottom left) shrink smoothly as the main camera takes over. (c) Main camera view is visualizing the event on the entire screen space.

Scan for a video!

Result Presentation

Since the split-screen effect is highly dynamic, the results are best visualized inside a video. It shows the user engaging with an event by moving through all three phases of the algorithm. The demo is presented in the 3D environment of the Sailing+ application but the camera movement mimics the Augmented Reality behaviour of a user with a phone camera.

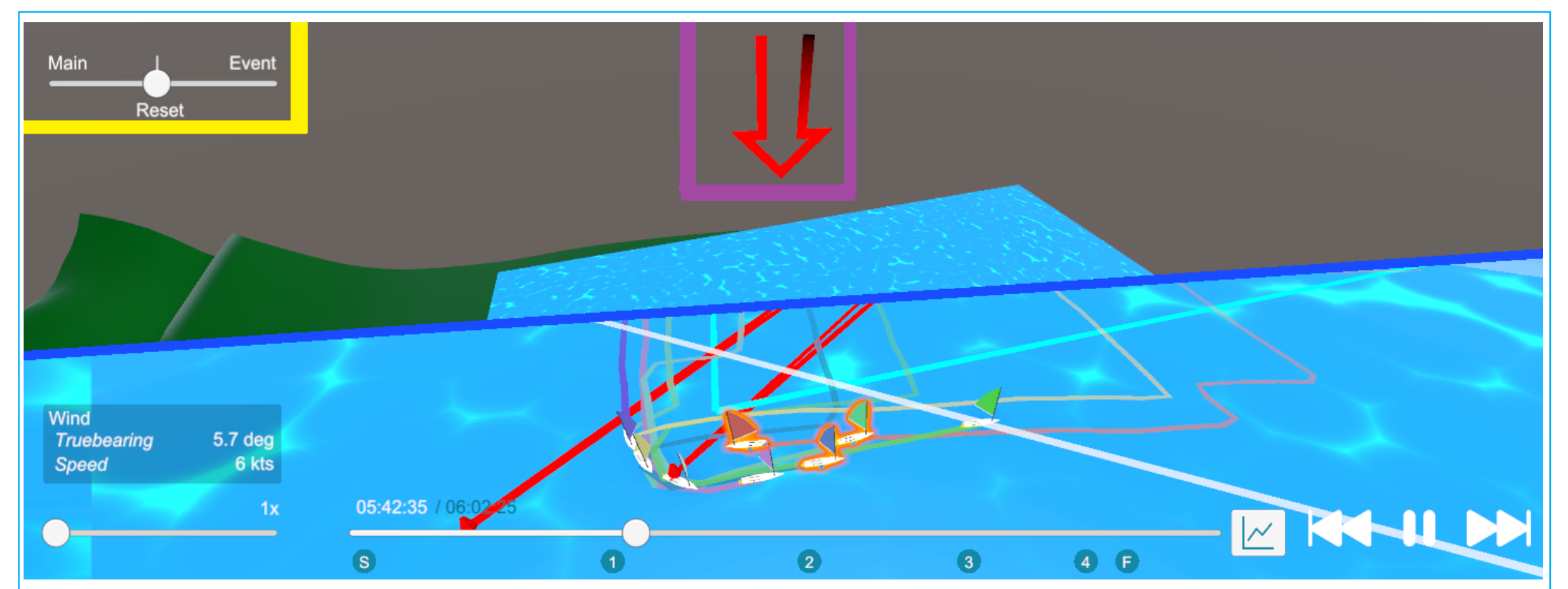
Figure 8: An arrow (highlighted with purple, top middle) is placed on the main camera view and points towards the event which in this case is at the bottom. The slider (highlighted with yellow, top left) is used to increase or decrease the camera views image, depending on the direction of the camera they would rather see (main to the left or event to the right). The reset button in the middle will revert the slider to its original position.