Location-Aware Adaptation of Augmented Reality Narratives (Supplementary Material)

WanWan Li*
wli17@gmu.edu
George Mason University and University of South Florida
Fairfax, Virginia, USA

Changyang Li*
cli25@gmu.edu
George Mason University
Fairfax, Virginia, USA

Minyoung Kim
mkim229@gmu.edu
George Mason University
Fairfax, Virginia, USA

Haikun Huang
hhuang25@gmu.edu
George Mason University
Fairfax, Virginia, USA

Lap-Fai Yu
craigyu@gmu.edu
George Mason University
Fairfax, Virginia, USA

CCS CONCEPTS

• **Computing methodologies → Mixed / augmented reality**.

KEYWORDS

interactive narratives, augmented reality, storytelling, path generation

ACM Reference Format:
WanWan Li*, Changyang Li*, Minyoung Kim, Haikun Huang, and Lap-Fai Yu. 2023. Location-Aware Adaptation of Augmented Reality Narratives (Supplementary Material). In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544548.3580978

1 IMPLEMENTATION DETAILS

Our synthesized navigation graphs can be employed for delivering interactive narrative experiences using a Microsoft HoloLens 2 augmented reality (AR) headset. We extended the HoloLens 2 with a GPS receiver and sunglasses (Figure 1) to facilitate tracked outdoor navigation for two reasons: (1) the display brightness of HoloLens 2 is usually insufficient for outdoor AR experience during a shiny day; (2) there is no embedded GPS sensor in HoloLens 2.

We developed a Universal Windows Platform (UWP) Dynamic Link Library (DLL) plug-in to access location data from the GPS bluetooth receiver. Our AR application tracks the position of the player using GPS, displays story events at the assigned locations, and shows wayfinding hints to guide the player to walk through story branches. It also enables the player to specify what action to take at a story event, which determines the story branch to follow.

* W. Li and C. Li contributed equally to this work.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3580978

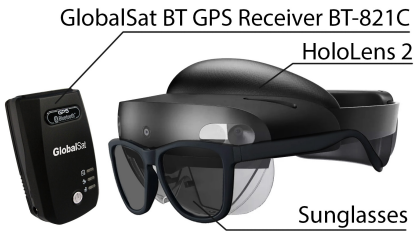


Figure 1: HoloLens 2 Extensions.

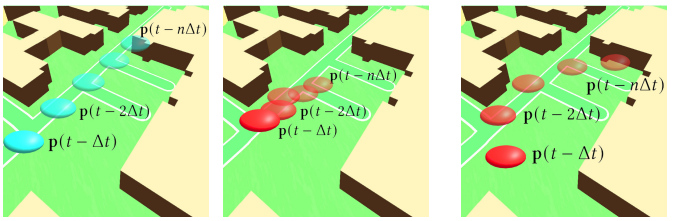
1.1 Search Step Self-Adaptation

We extend the optimization process with a self-adaptation process to perform local refinement near the end of the optimization in sampling the nagivation graph. The goal of the search step self-adaptation is to achieve the following effect: at the beginning, the sampled vertices are replaced by other vertices in the solution space globally. As time goes by, the new vertices are searched in nearby regions to perform local refinements. The self-adaptation steps include:

Step 1: for each vertex $v_i \in V$ in the navigation graph $G = (V, E)$, all of the other vertices $v_j \in V, i \neq j$ are sorted according to shortest distances $D(i, j)$ between vertex v_i and v_j. Sorted vertex indices for vertex v_i are listed in $U_i = \{u_1, u_2, ..., u_{|V|}\}$ s.t. $\forall j \in [1, |V| - 1] \rightarrow D(i, u_j) < D(i, u_{j+1})$. Put it another way, given any vertex index $u_j \in U_i$, v_{u_j} is the j^{th} nearest vertex of vertex v_i.

Step 2: create a new event node $s'_i = (e_i, l'_i)$ with a new random location vertex index l'_i that is within the k^{th} nearest vertices of v_i and has not appeared in story events S s.t. $l'_i \in \{u_1, u_2, ..., u_k\} \subseteq U_i, l'_i \neq l_i \wedge \forall s_j \in S \rightarrow l'_i \neq l_j$, where k is an integer decay along with the number of iterations. At the beginning, $k = |V|$ and the sampled vertices are replaced by other vertices globally. As time goes by, k becomes a small number so that the new vertices are searched locally to perform local refinements. We empirically decrease k with respect to iteration time $t \in [1, t_{\max}]$:

$$k = \max(\left\lfloor \left(1 - \left(\frac{t-1}{t_{\max}-1}\right)^2\right)|V| \right\rfloor, k_{\min}), \tag{1}$$



(a) Steady Velocity (b) Unsteady Velocity (Unsteady Space Intervals) (c) Unsteady Velocity (Inconsistent Moving Directions)

Figure 2: The timestamp algorithm. (a) Timestamps representing the player's previous positions, $\mathbf{p}(t-\Delta t), \mathbf{p}(t-2\Delta t), ..., \mathbf{p}(t-n\Delta t)$. In this case the velocity is steady. Unsteady velocities due to (b) unsteady space intervals; and (c) inconsistent moving directions.

(a) Building (b) Park (c) Road

Figure 3: Ground truth zoning maps used to test our autoencoder.

where $t_{\max}$ is the total number of iterations. We empirically set $k_{\min} = 20$ to ensure that at least 20 nearest vertices can be selected randomly to update the solution locally by the end of the optimization. Note that the decay of k is quadratic instead of linear as this will result in a better solution empirically. This self-adaptation approach works extremely well on dense navigation graphs. For a dense graph, it requires more local searches near the end to find the optimized solution. However, for a sparse navigation graph, it is better not to use this self-adaptation method as it may get the optimizer trapped in a local minimum early.

1.2 Location-Aware Event Simulation

With the GPS bluetooth receiver, the player's location can be tracked in real-time. Another consideration in simulating the location-aware virtual events during the storytelling is that there is no need to visualize the virtual contents in an event unless the player is nearby and is moving towards the event location. However, as the location data gathered by the GPS receiver only includes the latitude and longitude information, it is hard to decide the player's front direction. Therefore, we propose a timestamp-based solution to calculate the player's steady velocity and update the front direction according to the player's velocity direction only when the velocity is steady. As shown in Figure 2, up to time t, the timestamps representing the player's previous positions are $\mathbf{p}(t-\Delta t), \mathbf{p}(t-2\Delta t), ..., \mathbf{p}(t-n\Delta t)$, which are recorded by a timer with a time interval Δt. Then the player's velocity $\mathbf{v}(t)$ is:

$$\mathbf{v}(t) = \frac{1}{n}\sum_{i=0}^{n-1}(\mathbf{p}(t-i\Delta t) - \mathbf{p}(t-(i+1)\Delta t)), \tag{2}$$

where n is the number of timestamps. The player's velocity $\mathbf{v}(t)$ is considered as steady when two conditions are met:

- *Steady Space Interval*: The distance between every two adjacent positions in timestamps is long enough, i.e. $\forall i \in [0, n-1] \rightarrow ||\mathbf{p}(t-i\Delta t) - \mathbf{p}(t-(i+1)\Delta t)|| \geq \Delta d_{\min}$, where $\Delta d_{\min}$ is the minimal space interval.
- *Consistent Moving Direction*: The latest velocity $\mathbf{v}_0 = \mathbf{p}(t) - \mathbf{p}(t-\Delta t)$ and the earliest velocity $\mathbf{v}_{n-1} = \mathbf{p}(t-(n-1)\Delta t) - \mathbf{p}(t-n\Delta t)$ have a similar direction such that $\dot{\mathbf{v}}_0 \cdot \dot{\mathbf{v}}_{n-1} \geq 1-\epsilon$, where we set threshold $\epsilon = 0.1$.

Empirically, we set the number of timestamps $n = 5$, time interval $\Delta t = 1.5$s, and minimal space interval $\Delta d_{\min} = 0.2$m. As shown in Figure 2(b), the velocity is unsteady as it breaks the *Steady Space Interval* condition, there is not enough spatial interval between the timestamps. Similarly, Figure 2(c) shows the case that the *Consistent Moving Direction* condition is broken as the timestamps show that the path tends to move along a curve. Therefore, if the player's velocity is steady and the player is moving forward along a line on a road, and if the player is nearby an event object, that object will be activated. That object will appear in front of the player with a distance d calculated by the player's location $\mathbf{l}_p = (\phi_1, \lambda_1)$ and the event's location $\mathbf{l}_e = (\phi_2, \lambda_2)$ using the 'haversine' formula:

$$d = 2R\tan^{-1}\sqrt{\frac{\alpha}{1-\alpha}}. \tag{3}$$

Here, ϕ is the latitude and λ is the longitude. We use $R = 6,371$km as the Earth's radius. We calculate α as:

$$\alpha = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_1\cos\phi_2\sin^2\left(\frac{\Delta\lambda}{2}\right), \tag{4}$$

where $\Delta\phi = \phi_2 - \phi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$.

2 ADDITIONAL DETAILS OF EXPERIMENTS

2.1 Adapting Augmented Reality Narratives

Most of our results are based on the *Detective AR* story shown in Figure 5 (a). The story branches are depicted with detailed explanations. The events are accompanied by the main characters and their dialogues. The arrows refer to the player's choices at the events which will determine the story branch the player goes through. Every event is colored according to its compatible zone type. For example, Event A (in dark blue) is supposed to happen in a teaching zone.

2.2 Autoencoder Zoning

We provide an alternative, learning-based approach to infer the probabilities of locations belonging to different zone types, which can be used for evaluating the location compatibility score function. More specifically, we use an autoencoder to infer zone types. We test the accuracy of our autoencoder using a cross-validation approach. First, the ground truth is manually created, where the zones are labeled according to the 3D view on satellite Google Map as shown in Figure 3.

Figure 4 shows the zone types inferred by our trained autoencoder together with the street view images at some sample points.

(a) Regions with building features (b) Regions with park features (c) Regions with road features

Figure 4: Zone types inferred by the autoencoder. The color intensities indicate how likely the locations belong to different zone types.

Our synthesized navigation graph using the inferred zoning maps is based on a modified version of the *Detective AR* story as depicted in Figure 5 (a), in which we change the zone types and event location descriptions accordingly.

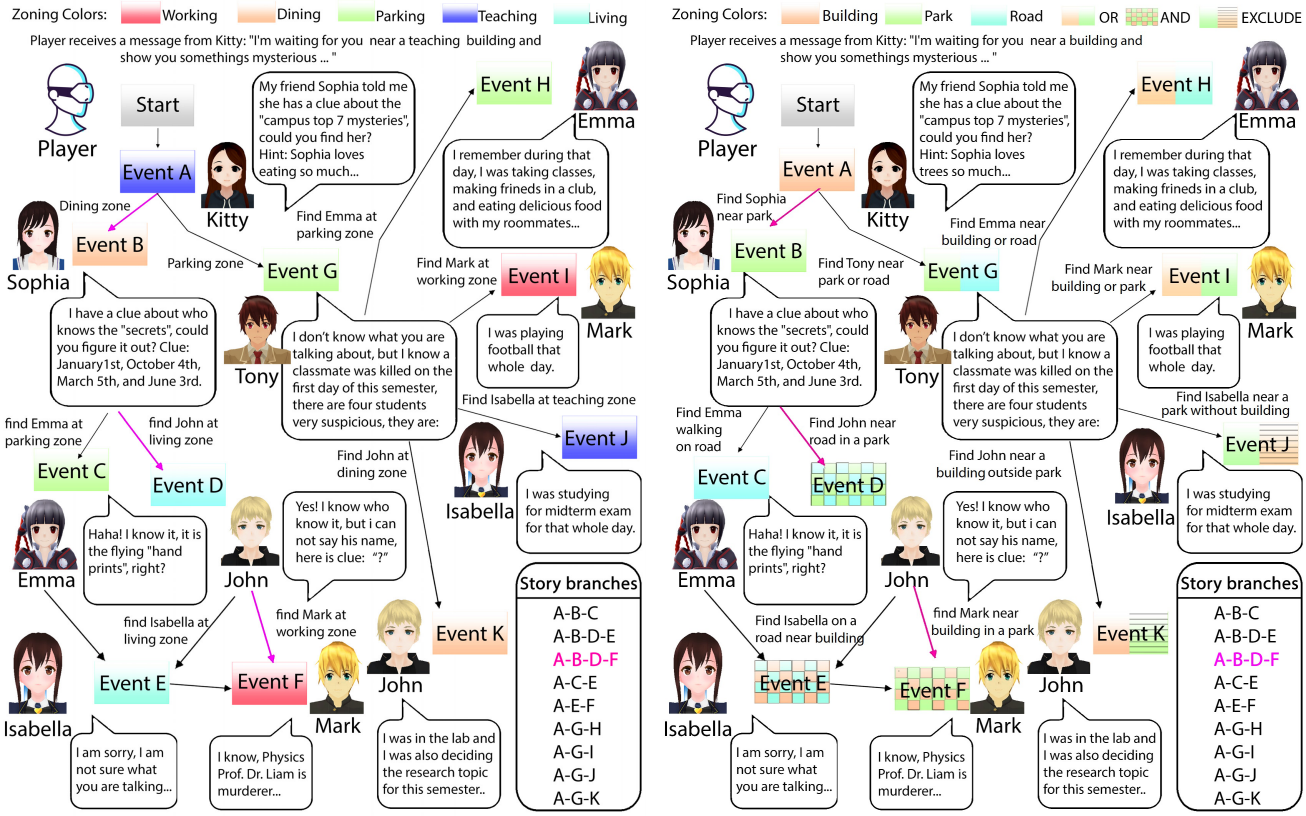
2.3 Scalability

For the two additional stories of larger scales, *Dinosaur World AR* is shown in Figure 6, and *Exterior Star War AR* is shown in Figure 7.

3 ADDITIONAL DETAILS OF COMPARISON WITH MANUAL DESIGN

The general feedback from the 15 designers about manually distributing story events is included below:

(1) One designer thinks the design task wasn't very difficult, but there was a learning curve to figure out what pin (event) placements are correct and reachable.
(2) One designer said that it was hard to find the optimal path among all the possible path, and the process took some time. Also, it was hard to find out unnecessary routes, and misplaced pins and redundant paths took efforts to fix.
(3) One designer thinks that it wasn't difficult to find the correct spot to put each pin to minimize the length. The task was interesting and straightforward.
(4) One designer thinks what was difficult was predicting what the player would be doing during the placement task.
(5) One designer thinks the task is intuitive for a designer. To be more specific, it is easy to learn how to use the tool. The analysis on mismatches and walking distance provided by the tool help create a satisfactory design.
(6) One designer complains about the need to figure out which pins cause problems (referring to locations that are not reachable on the map). If the tool can provide some hints about the problems of a design, it would be helpful for the designer.
(7) A designer pointed out that the length of the path is much shorter if the pins are all placed in the same general location. This is convenient when a short path is wanted, but it may be too repetitive for the player to keep visiting nearby locations. Also, it was fun to distribute story events on the map.
(8) One designer thinks the design task wasn't very difficult. However, it takes time to figure out and trace the paths.
(9) One designer finds that the pin placement a bit nonintuitive with regards to where the pins exactly refer to on the map.
(10) One designer thinks it is hard as there are too many possible locations to consider. There is no straight answer to this problem. A lot of time is spent on trial and error.
(11) One designer thinks that the task is difficult because even though it is easy to keep adjacent events close to each other, it is hard to optimize the overall placement, especially when an event can be connected with multiple other events.
(12) One designer thinks that it is not hard to shorten the paths at the beginning. However, it is hard to know if it is close to the optimal solution. Placing pins on a compatible zone (with the same color as the pin's) is easy, but finding the shortest paths is hard due to too many path and location possibilities.
(13) One designer thinks that the task is difficult as it takes much effort to match the story events with their compatible zones in the scene and also to minimize the total distance.
(14) One designer thinks that it is difficult as both color (zone) matching and story tree matching need to be considered.
(15) One designer thinks that the operation is difficult since it is hard to know if moving a pin to another location improves the solution. The designer needs to compute the average distance and the standard deviation in distance every time a move is applied to make sure that the move makes sense.



(a) Original *Detective AR* (b) Modified *Exterior Star War AR*

Figure 5: (a) A story, *Detective AR*, used as our illustrative example; (b) The *Detective AR* story modified for the autoencoder-based synthesis. *OR* (denotes $Z_1 \vee Z_2$), *AND* (denotes $Z_1 \wedge Z_2$) and *EXCLUDE* (denotes $Z_1 \wedge \overline{Z_2}$) in the legend are used for illustration convenience.

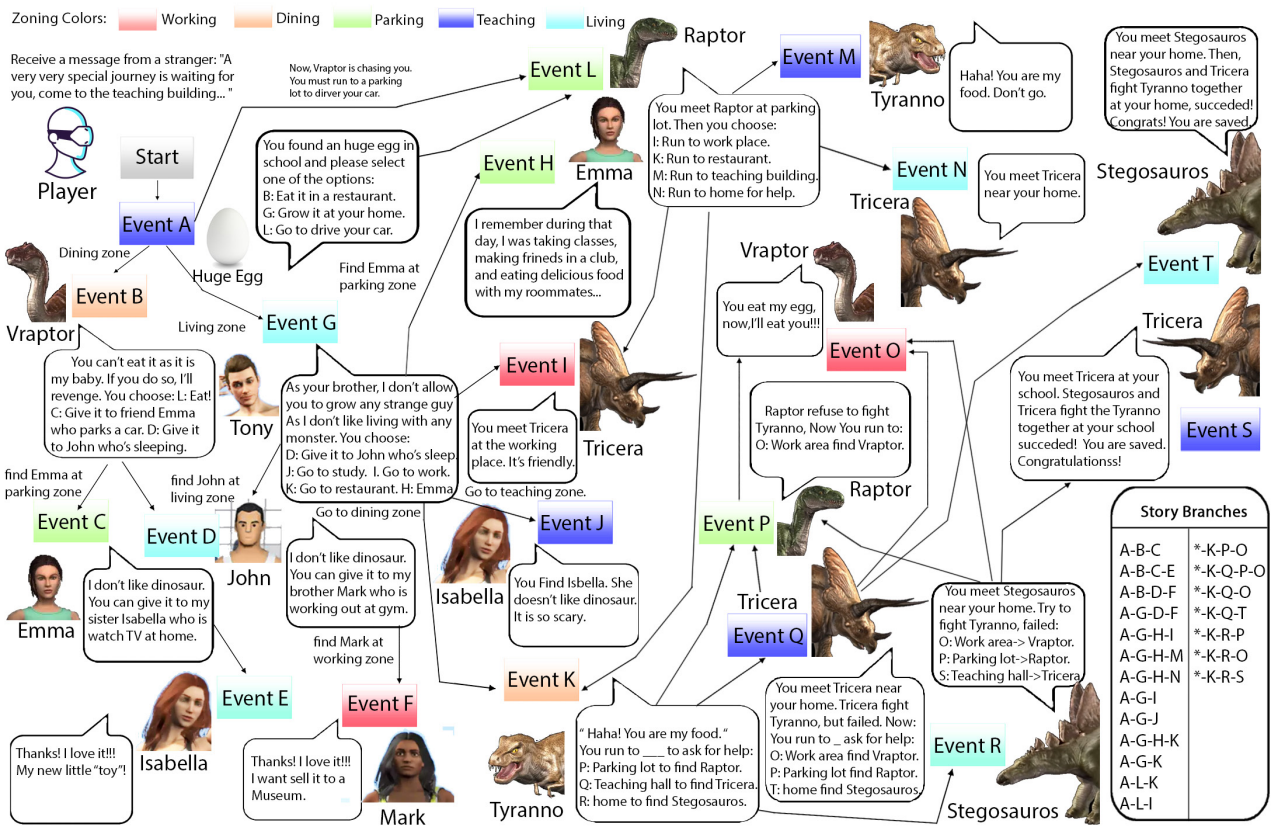


Figure 6: Medium-scale story example, *Dinosaur World AR*. Note that in the story branch description, the prefix * indicates all possible matches. For example, *-K-P-O refers to A-G-H-K-P-O, A-G-K-P-O, and A-L-K-P-O.

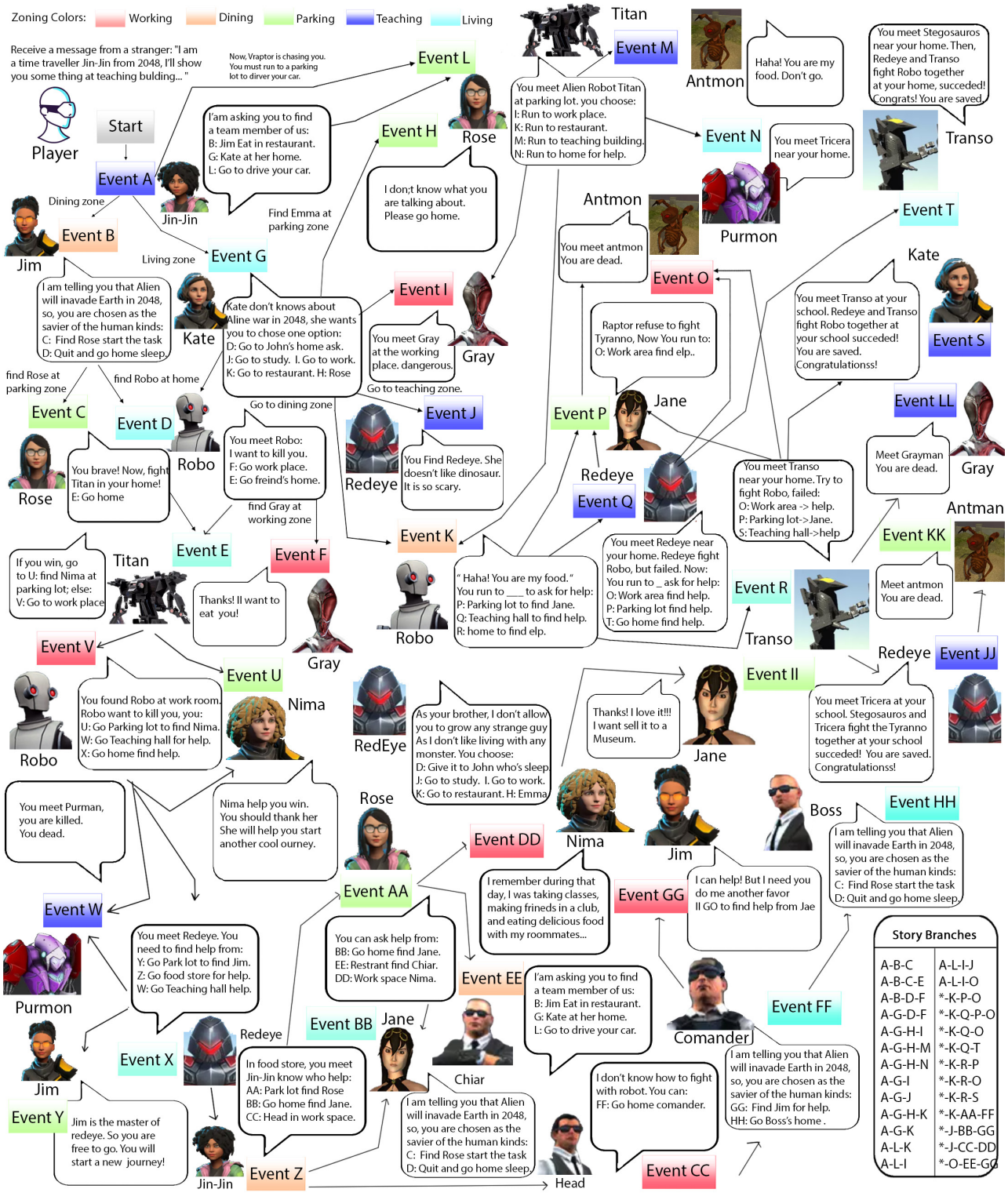


Figure 7: Large-scale story example, *Exterior Star War AR*.