Evaluating the Authoring Complexity of Interactive Narratives with Interactive Behaviour Trees Supplementary Document

1. INTERACTIVE BEHAVIOR TREES

Interactive Behavior Trees (IBT's) address challenges in traditional Behavior Tree representations by facilitating free-form user interaction and state persistence. IBT's, as illustrated in Fig. 1(a) are divided into 3 independent sub-trees that are connected using a Parallel control node. An IBT $\mathbf{t}_{\text{IBT}} = \langle \mathbf{t}_{\text{ui}}, \mathbf{t}_{\text{state}}, \mathbf{t}_{\text{narr}} = \{\mathbf{t}_i^{\text{arc}} | \mathbf{t}_1^{\text{arc}} \dots \mathbf{t}_m^{\text{arc}}\}, \beta \rangle$ where: (1) $\mathbf{t}_{\text{narr}}$ is the narrative definition with modular story arcs $\{a_i\}$, each with their own independent subtree $\{\mathbf{t}_i^{\text{arc}}\}$. (2) $\mathbf{t}_{\text{ui}}$ processes the user interactions. Fig. 1(b) illustrates the story subtree. (3) $\mathbf{t}_{\text{state}}$ monitors the state of the story to determine if the current story arc needs to be changed. Fig. 1(b) illustrates the story subtree. (4) β stores the state of the story and its characters.

Story Subtree. $\mathbf{t}_{\text{narr}}$ is responsible for handling the narrative progression and is further subdivided into subtrees that represent a separate story arc. Fig. 1(b) provides an example of $\mathbf{t}_{\text{narr}}$ while Fig. 1(c) illustrates each arc definition $\mathbf{t}^{\text{arc}}$ which is encapsulated as a separate subtree. This introduces an assertion node which is checked at every frame whether the current arc is still active before proceeding with its execution. This minor extension to the story arc definition allows the story to instantaneously switch arcs at any moment in response to the user's interactions.

MonitorUserInput Subtree. $\mathbf{t}_{\text{ui}}$ monitors the different interactions that are available to the user and can be easily changed depending on the application or device. Once an input is detected, it sets the corresponding state in the blackboard β which is queried by $\mathbf{t}_{\text{state}}$ to determine the current state of the story, and the active story arc. Since $\mathbf{t}_{\text{ui}}$ is executed in parallel with the other subtrees, we are able to immediately respond and register the interactions of the user and use it to influence the narrative outcome. Fig. 1(d) illustrates an example.

MonitorStoryState Subtree. $\mathbf{t}_{\text{state}}$ contains separate subtrees for each story arc which checks if the precondition for the particular arc is satisfied. If so, β is updated to reflect the newly activated story arc which is used to switch the active story in $\mathbf{t}_{\text{narr}}$. Fig. 1(e,f) illustrates $\mathbf{t}_{\text{state}}$ and a subtree used for checking the preconditions for the Honeypot story arc. It may be possible for the preconditions of multiple story arcs to be satisfied at any instance, in which case the story arcs are activated in order of priority (the order in which they appear in $\mathbf{t}_{\text{narr}}$. It is also possible for multiple story arcs to be active simultaneously if they are operating on mutually exclusive characters and objects.

Message Passing and State Persistence. The overall design of the IBT results in three subtrees that execute independently in parallel with one another. A blackboard β is used to store the states of the characters and the story and is used to communicate between the subtrees and also to maintain state persistence. $\mathbf{t}_{\text{ui}}$ updates β when any input signal is detected. For example, if the user places a honeypot sticker, it triggers a honeypot in the world and sets the appropriate flag in β . Tree $\mathbf{t}_{\text{state}}$ monitors β to check if the preconditions of a particular story arc is satisified, and updates the current arc. Finally, each arc subtree in $\mathbf{t}_{\text{narr}}$ checks if it is the current active arc before continuing. Also, the user input and the narrative execution can update story and character state to influence the progression of the narrative at a later stage. For instance, the bears may remember how the user interacted with them at the beginning of the story and this may impact their behavior later on.

2. AUGMENTED REALITY FRAMEWORK

Our framework models a virtual feedback loop between a narrative and a user. It provides a narrative to the user while at the same time captures the user's reactions and interactions and integrates them into the narrative. We define an interaction vocabulary, which models how the user can interact with the narrative.

Our approach is centered around the user and the interactions available to the user. The user points the mobile device at the marker. The interactive narrative is displayed on the device screen, directly placed inside the captured real environment. A variety of sensors is used to monitor how the user interacts with the narrative. The low-level input data of the sensors is interpreted in an interaction vocabulary and reported to the behaviour tree controlling the narrative. The behaviour tree reacts to the events reported by the interaction vocabulary and changes the story plot accordingly. This allows the user to interact freely with the

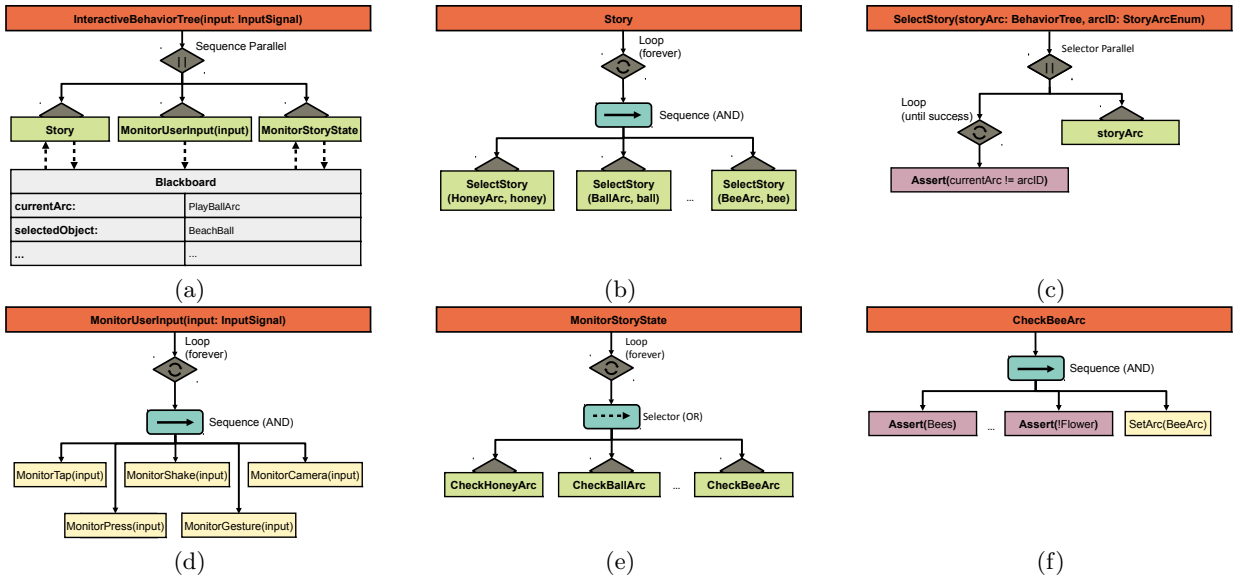

Figure 1: (a) Design formalism of Interactive Behavior Trees (IBT's) with decoupled specification of user input, narrative definition, and the impact of user input on story state. (b) Narrative subtree with modular story arcs. (c) Each story arc definition is encapsulated in its own independent subtree which first checks if this is the current active arc before proceeding with the narrative execution. (d) Subtree to monitor user input. (e) Subtree that changes story state based on user input which triggers branches in story arc. (f) An example subtree from (e) which checks if all the preconditions for a particular story arc are satisfied before setting it as the current active arc.

narrative and branch into new subtress at any point in time.

2.1 Architecture Overview

Our architecture is composed of five principal components. The user provides our platform with the data necessary to create an interactive experience. A set of sensors monitors the user and his interactions with the mobile device at all times and forwards them to the interaction vocabulary. The interaction vocabulary is processed in a next step, interpreting and analysing the incoming low-level input from the sensors. These low-level input events are mapped to higher level interactions. Additionally, the sensor data from the camera is used in a camera pose estimation step to compute the pose of the mobile device. The pose, together with the high-level events is then forwarded to the behaviour tree. The BT checks the input-events and adjusts the plot of the narrative being shown accordingly. Finally, the currently unfolding story is presented by a game engine, which is controlled by the BT.

2.2 Augmented Reality

In an AR application, the real, physical world is merged with a virtual world. In see-through AR, both worlds are merged on the screen of a mobile device, capturing the real world with it's camera and overlaying a virtual, rendered world.

AR applications benefit from intuitive and versatile input mechanisms, that is, the user can seemingly physically interact with the virtual world. For see-through AR applications, the physical movement of the mobile device already provides a straight-forward and easy understandable interface to interact with the application.

Furthermore, in an AR application, not only can the user physically interact with the virtual characters but the virtual characters can interact with the physical world as well. Using latest technologies, such as Simultaneous Localization and Mapping (SLAM) [4] or Vuforia Smart Terrain, the environment is captured and integrated in the application. For example, the user can place a box in a virtual canyon to clear a path for the virtual characters to get over the canyon unharmed.

Implementation. AR technology is available in a broad variety. Location-based and city-wide AR applications that employ GPS information, accelerometer, and gyroscope data to locate the user, see-through AR applications on mobile devices that use just the camera, and projection-based AR that project virtual content onto real geometry using static calibrated projectors are the most common applications of AR. In addition to the type of AR, the implementations also differ in that some methods track a 2D marker image [1] and other track 3D geometry [2]. In our specific case, the stability and robustness of the AR implementation is the most important factor, as an unstable implementation will completely break immersion.

As fiducial and natural image marker based implementations have been around longer than the alternatives, they are the safe route when aiming for stable tracking [6]. We decided to use the natural image marker based approach provided by Qualcomm's Vuforia [3]. Vuforia performs image registration by extracting image features, such as SIFT (cite Lowe, 2004) features and estimates the camera location and pose in real-time.

2.3 Interaction Vocabulary

1. **Movement**. is used to make the narrative react to the position of the user. Camera input is interpreted using CPE, giving us an accurate estimation of the device position in the virtual world. This allows us to make characters look at the user, throw balls to the user or, in general, interact with the users position in the virtual world.

2. **Look**. is used to interact with the narrative based on the angle at which the user looks at the scene. The direction of the device is again estimated using CPE and is done in the same pass as the position estimation.

3. **Taps**. are very short series of touch events. The user can tap on an object to initiate an interaction. Low-level touch events are recorded over a certain amount of time and constantly analysed by the interaction vocabulary. A series of touches is interpreted as a tap, if the duration between the start of the series and the end of the series is smaller than a specified threshold value and if the delta position between each consecutive pair of events and the start and end event is smaller than a tolerance threshold value.

4. **Longpresses**. are longer series of touch events. The user can press on objects for a longer time to influence them in a different way than when tapping them. As with taps, long presses are discovered by looking at a series of low-level touch events over a certain amount of time. A series of touches is interpreted as a long press, if the duration between the start of the series and the end of the series is larger than a specified threshold value and if the delta position between each consecutive pair of events and the start and end event is smaller than a tolerance threshold value.

5. **Shaking**. can be initiated by shaking the device in any direction. Shaking is implemented by approximating the integral over the amplitude of the accelerometer events in quarter second frames and comparing the resulting value with a threshold value. This will trigger start and end shake events with an accuracy of a quarter second.

6. **Gestures** are used to express desires and wishes to characters. We use the HyperGlyph library to detect single stroke gestures. When compared to a $1 recognizer [5], HyperGlyph rejects and confirms matches more accurately.

3. EXAMPLE OF INTEGRATING USER INPUT

One example that was shown in the naive approach towards state persistance was the difference between the bears playing ball with the user or without him. Fig. 2 and Fig. 3 show the corresponding BT's of those subtrees.

We wanted to have a BT that is designed in a general way to make it possible to monitor the whole interaction of this game. Fig. 2 shows the design of such a BT, where two characters play ball with each other. First, we need to check which character currently holds the ball to decide which one throws the ball. The other character will catch it, which is directly done in the "ThrowTo" node. After completing this, the BT will simply loop again over those interactions until the ball was thrown n times. This number can be directly assigned by the developer.

To integrate the user in this game of playing ball, we need to modify the BT. There are two challenges we have to take care of:

- **User as a participant:** The user should be seen as an active participant that the character can throw the ball to.
- **Wait for user interaction:** If the user has the ball, the characters need to wait for the user to throw it back into the scene.

We solved the first challenge by using a probability selector. The character currently holding the ball can throw the ball either to the other character or to the user. By assigning probabilities to each possible branch, the character will randomly throw to the user or to the other character. Using this, it is possible for all characters in the game to see the user as simply another character in the story.

The second challenge can easily be solved by first checking if either of the two characters hold the object and otherwise we will enter the subtree **UserThrowBall** in which we will wait until the user reintroduces the ball into the story and one of the characters picks it up.

Fig. 4 shows another small example of a subtree in our story definition. The subtree **Talking** lets the bears talk to each other. During this conversation the second bear asks for a beach ball. The first bear can not help him and the second bear does not want to talk anymore. He is sad and leaves the first bear, while the other one is also sad, because he couldn't help his friend. Therefore, he decides to ask the user for help. After this subtree the story will not continue until the user interacts with them by e.g. throwing the ball into the story or spawning the bees. This example shows very nicely how the user can be actively involved as a participant to move the story forward. As described in the paper the Interactive Behavior Tree (IBT) monitors the user interaction in a separate subtree called **MonitorUserInput**, which is able to change the global state depending on the user input. Using the global state the **MonitorStoryState** subtree decides which story arc should be executed by modifying the variable **currentArc** in the blackboard. The Story subtree will change the narrative according to this variable. We will shortly explain how the story can continue depending on the user.

1. **Throw the beach ball to the first bear.** If the ball was thrown to the first bear, we will enter the story arc PlayBallArc. The first bear picks up the ball, gives it to the second one and both are happy. They will start playing ball with the user.
2. **Throw the beach ball to the second bear.** If the ball was thrown to the second bear, the story enters the story arc SadBearArc. The second bear will pick

it up and will thank the user. The first bear is sad, because he couldn't give him the ball. In the end, they are both friends again and also start playing ball with the user.

3. **Throw the soccer ball.** Throwing the soccer ball leads to executing the WrongBallArc. The bears are not interested in the soccer ball. Therefore, the first bear will simply shake its head to indicate that this is the wrong ball and continues to ask the user for help.

4. **Spawn Bees.** Spawning the bees results in executing the BeesArc. Upon seeing the bees, the bears will panic and try to escape the bees. The only way to distract the bees is to spawn flowers. The bears will stop panicking and the second bear will continue asking the user for help.

5. **Place the honey pot.** This leads to enter the HoneyArc. The bears simply love honey. That is why they will quickly forget about the ball and start eating honey. If the user takes away the honey pot, both bears will be sad and the second bear continues asking the user for help. Otherwise, the story will end after both bears have eaten for a while.

4. REFERENCES

[1] F.-e. Ababsa and M. Mallem. Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems. In *ACM SIGGRAPH VRCAI*, pages 431–435, 2004.

[2] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on*, pages 117–120, Sept 2008.

[3] Qualcomm. Vuforia Developer SDK, 2010.

[4] G. Tuna, K. Gulez, V. Gungor, and T. Veli Mumcu. Evaluations of different simultaneous localization and mapping (slam) algorithms. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 2693–2698, Oct 2012.

[5] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[6] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), Dec. 2006.

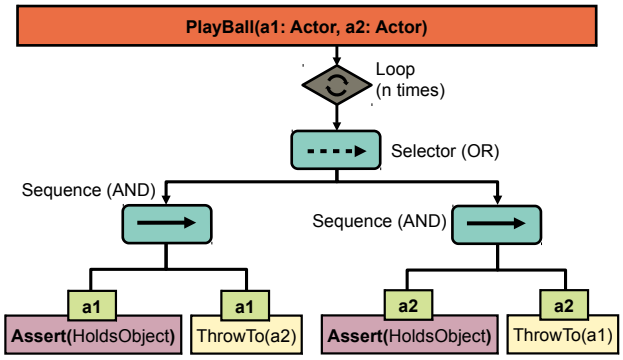

Figure 2: PlayBall. A simple BT that authors a game of playing ball between two characters.

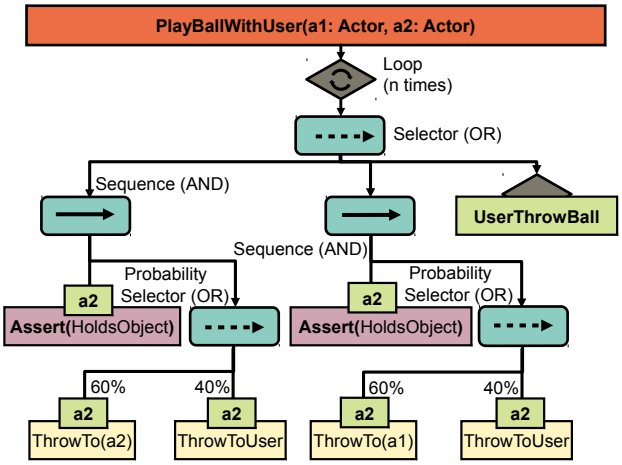

Figure 3: PlayBallWithUser. A modification of Fig. 2 in which the user is integrated into the game.

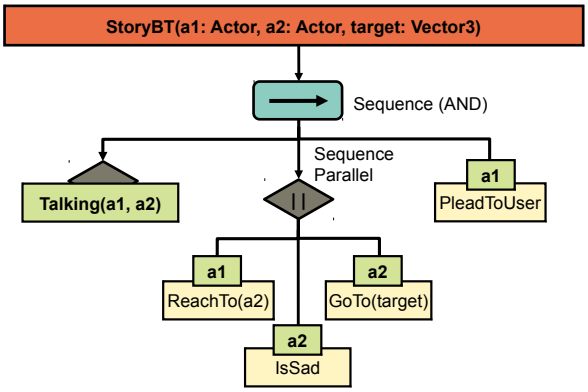

Figure 4: Example subtree of the story definition. This BT shows another small example of a subtree in our story definition. The two bears are talking to each other and the second bear gets sad. The first bears asks the user for help.