Interactive drama

- Interactive drama combines autonomous characters and drama management to create first-person story-worlds
- Problems that have to be solved
  - Communication with characters (natural language, gestures, ...)
  - Maintaining a story structure (causally unified chain of events with closure)
  - Believable characters (personality rich, emotional, lifelike behavior)
- Interactive drama requires unifying the character and story view
- In this lecture we’ll talk about
  - The overall architecture and approach of Façade
  - The ABL programming language

Façade
Overview of approach

How is Façade like a simulation

- On a moment-by-moment basis, Façade is open-ended
- You are free to move in 3D anywhere you want at any time, to pick up and use objects, to say anything you want at any time (by typing)
- Grace and Trip are autonomous characters that can perform dramatic dialog in multiple ways and places, have goals, including personality and emotion behaviors
- Grace and Trip are directly, immediately responsive to your dialog and actions – local agency, character
  - Requires a large collection of behaviors to achieve interesting local agency

How does Façade have narrative structure

- Façade has collections of coordinated character behaviors called story beats
- The current beat supplies Grace and Trip each with a rich set of additional behaviors that combine with their already existing innate behaviors, giving them focused goals to pursue for the next ~60 seconds of the simulation
- Each beat is designed to attempt to accomplish a nugget of narrative action, for example, forcing the Player to choose sides on an issue, play a psychological headgame with Player to increase tension, or reveal important information to the Player
- A beat manager is responsible for deciding how to sequence the beats
Hybrid between simulation and narrative structure

- On a moment-by-moment basis, Façade is as open-ended as a simulation.
- But every minute or so, the drama manager reprograms the simulation in order to give the overall experience some narrative shape.
- Somewhat akin to levels in a more traditional game, except a new level happens every minute or so.

Façade architecture

Natural language understanding
Two phases of the Façade NLU

NLU: Surface text to discourse acts

- Surface text
- Discourse acts (~25)
- Conversation management: Discourse acts to reactions

- Context: Affinity Game
- Context: Global

Focus on what language does: Pragmatics

- “it’s not a problem”

- Syntax – produce a parse tree of the sentence

- Semantics – represent the formal meaning of the sentence

- Pragmatics – the conversational “move” of the sentence

Example discourse acts

<table>
<thead>
<tr>
<th>Representation of Discourse Act</th>
<th>Pragmatic Meaning of Discourse Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{DAAgree} (?char)</td>
<td>Agree with a character (e.g. “certainly”, “sure thing”, “I would love to”)</td>
</tr>
<tr>
<td>\text{DADisagree} (?char)</td>
<td>Disagree with a character (e.g. “No way”, “Fat chance”, “Get real”, “Not by a long shot”)</td>
</tr>
<tr>
<td>\text{DANegExcl} (?char)</td>
<td>A negative exclamation, potentially directed at a character (e.g. “Damn”, “That really sucks”, “How awful”, “I can’t stomach that”)</td>
</tr>
<tr>
<td>\text{DAThank} (?char)</td>
<td>Thank a character (e.g. “Thanks a lot”)</td>
</tr>
<tr>
<td>\text{DAAlly} (?char)</td>
<td>Ally with a character (e.g. “I like you”, “You are my friend”, “I’m here for you”)</td>
</tr>
<tr>
<td>\text{DAExplain} (?char ?adj)</td>
<td>Explain something simple about a character (e.g. “You’re afraid”, “Trip is controlling”, “Grace is angry”)</td>
</tr>
<tr>
<td>\text{DAExplainRel} (?char1 ?char2 ?rel)</td>
<td>Explain about a relationship between characters (e.g. “Grace doesn’t love Trip”, “Trip is cheating on Grace”)</td>
</tr>
</tbody>
</table>
Social games

- Affinity game
  - Player must take sides in character disagreements

- Hot-button game
  - Player can push character hot-buttons (e.g. sex, marriage) to provoke responses

- Therapy game
  - Player can increase characters' understanding of their problems

- Tension
  - Not a game, but dramatic tension increases over time and is influenced by player actions (e.g. pushing character hot-buttons can accelerate the tension)

Multiple, mixable progressions

- Each social game, plus tension, forms a mixable progression

- A progression consists of
  - Units of procedural content (e.g. beats, beat goals)
  - A narrative sequencer that manages the progression and responds to player interaction

- Multiple progressions run simultaneously and can intermix

Façade’s social games

- Affinity game
  - Player must take sides in character disagreements

- Hot-button game
  - Player can push character hot-buttons (e.g. sex, marriage) to provoke responses

- Therapy game
  - Player can increase characters' understanding of their problems

- Tension
  - Not a game, but dramatic tension increases over time and is influenced by player actions (e.g. pushing character hot-buttons can accelerate the tension)
The progressions

- Beat sequencing (overall story + tension)
- Beat goal sequencing (affinity game)
- Global mixins (hot button game)
- Therapy game similar

- Beat manager
- Beat library
- Canonical beat goal sequence
- Handlers (ABL meta-behaviors) + discourse management
- Mixin library
- Handlers + discourse

Story AI: authorship and interaction

The **Enemy**

- Author has control but
  - All interaction paths must be pre-coded by author
  - Can only make very small stories
  - Bits of story can’t be incrementally added

Beat manager
Drama management

- Policy for “story piece” selection
- An alternative to explicitly coded links

Drama management design space

- What are the story moves?
- How is the desired story represented?
- What is the selection policy?
- When does selection happen?

Façade: Beat-based drama manager

- What are the story moves?
  Dramatic beats
- How is the desired story represented?
  Beats (declarative selection knowledge, procedural performance knowledge)
- What is the selection policy?
  Maintain probability distribution over potential next beats
- When does selection happen?
  When a beat completes or aborts
Dramatic beats

- The beat is the smallest unit of value change
  - Coordinated action which moves story forward
- Story values – properties of characters and relationships that change over the story
  - Examples: anger, love, trust, hate, tension
- Activity without value change is not dramatic action

Beat example

- Beats become architectural entities
  - Declarative knowledge to sequence beats
  - Procedural knowledge to coordinate characters
  - The unit of meaningful player interaction

Beat: Introduce Grace’s buried desire to be an artist

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>story at tension 1 and after greeting and activity is neutral</td>
<td>Grace complains about her decorating Trip insists it looks great They force player to take a side</td>
</tr>
</tbody>
</table>

Beat manager architecture

- Language
  - Declarative knowledge influencing beat sequencing
  - Specify desired dramatic arc(s)
- Sequencer
  - Probabilistic agenda – draw from a changing beat distribution
Authoring content within Façade

The atom of performance
- Joint dialog behaviors form the atom of performance
- Façade consists of ~2500 joint dialog behaviors
  - Each 1-5 lines of dialog long (5-20 secs)
  - System sequences these, including transitions between
  - Most are interruptible
  - JDBs use ABL’s joint intension framework to coordinate performance

Types / uses of joint dialog behavior
- Beat goals and beat mix-ins
- Global mix-ins
- Autonomous mix-in behaviors
Types / uses of joint dialog behaviors (1)

- Beat goals and beat mix-ins
  - Progression of a specific topic – “beat”
    e.g., arguing about Grace’s decorating; showing off while fixing drinks
  - ~10–100 jdb’s per beat – 66% of the 2500 total
  - Sequenced in response to the player’s action by the beat’s meta behaviors and reaction proposers

---

Canonical beat goal sequence

---

Beat goal behaviors

---

Beat mix-ins
### Types / uses of joint dialog behavior (2)

- **Global mix-ins**
  - Progression of a global topic category
    - e.g., divorce, sex, the view, the wedding picture
  - 1 jdb each – 33% of the 2500 total
  - Sequenced in response to the player’s action by the global meta behaviors and reaction proposers

### Types / uses of joint dialog behavior (3)

- **Autonomous mix-in behaviors**
  - Physical behaviors in parallel with beats and global mix-ins, with a bit of dialog
  - Occasionally mixes in a jdbp – 1% of 2500 total
    - e.g., staging, fixing drinks, eightball
Bag of beats

- 27 beats total; ~15 seen in one runthrough
- Some beats have rich internal variation
  - Each jdbp may have 2-3 affinity X 2 tension variations, for a total of 100+ jdbp’s in a beat
- Sequenced by beat manager to match global tension arc
- Easy to add or take away beats

Collection of global mix-ins

- 3-tiered progressions – digging deeper
- ~20 global topic categories
  - objects: furniture, wedding pic, brass bull, view, etc.
  - satellites: marriage, divorce, sex, therapy, etc.
- Variation
  - 2-3 affinity X 2 tension
- First half of drama
- Slew of generic deflects and recoveries

Special beats

- Recap beats
  - Crisis – halfway though drama
  - Revelation Buildup – just before climax
- Therapy Game beat
  - Crisis -> Climax
  - Large collection (~150) of jdbp’s, play 25 (~5 min)
  - Direct access + custom search-based sequencing
Artistry and crafting

- Designing joint dialog behavior pairs
  - Beat goals and beat mix-ins
  - Global mix-in progressions
  - Autonomous mix-in behaviors

All of the above must INTERMIX coherently – a real struggle to achieve

Artistry and crafting (part 2)

- Coverage of player expression: abstraction and mappings
  - Best short list of parameterized discourse acts (~25)
  - Surface text -> discourse act mapping
  - Competing DA -> Reaction mappings (proposers and context priority mappers)

Artistry and crafting (part 3)

- High level beat progression
  - 27 beats
  - Some are special recap beats

- Writing good dramatic dialog

- Emotive procedural animation
ABL: A Behavior Language

New programming constructs for believable characters

- In creating Façade, we developed programming constructs for believable characters
- We created a new language to support these constructs
  A Behavior Language—ABL
  - Based on the CMU Oz-project language Hap
  - Reactive-planning: characters organized as goals and behaviors
  - Lessons from these constructs can be generalized beyond ABL
    A different way of thinking than imperative languages
    (e.g. C++, Java)

Façade character requirements

- Moment-by-moment believability
  Body movements, facial expression, behavior mixing
- Tightly coordinated action
  Characters work closely together to perform story
- Conversational behavior
  Longer-term, non-linear dialog flow that preserves reactivity
Code support for character requirements

- Goals and behaviors
  Sequencing + reactivity, behavior mixing, hierarchy
- Joint goals and behaviors
  Protocol supporting multi-character teamwork
- Meta-behaviors
  Canonical behavior sequences are modified by player interaction

Façade architecture

A behavior-based agent
Features of our behavior-based agents

- Characters organized as goals and sequential & parallel behaviors
- Joint (synchronized) goals and behaviors
- Reflection (meta-behaviors)
- Generalization of sensory-motor connections
- Multiple named working memories
- Atomic behaviors (useful for atomic WM updates)

Behaviors

Behaviors consist of steps
- Similar to the scripts or functions associated with FSM states, but
- Can be parallel as well as sequential
- Mix together as multiple behaviors are pursued

Behaviors are chosen to accomplish a goals
- Similar to function calls but
- Are dynamically chosen given current game conditions
- Can be re-chosen if the first choice doesn’t work out

Example behaviors

To answer the door:
1. Wait for knock
2. Sigh
3. Open the door
4. Greet the guest

If there is knock and the door is too far away, yell for guest to come in.
**Steps**

- Subgoal – chooses behaviors
- Act – does a physical act in the world
- Mental act – a bit of computation (e.g., change memory)
- Wait – used with conditions to accomplish demons

*All steps succeed or fail*

*Behavior finished when all steps succeed or one step fails*

*Behavior success and failure propagates up ABT*

**Continuously monitored conditions**

- Success tests – spontaneously make a step succeed if test is satisfied
- Context conditions – spontaneously make a behavior fail if test is satisfied
- Makes behaviors immediately reactive to changes in the world

**Success and failure propagation**
**Example: Sequential behavior + subgoaling**

```plaintext
sequential behavior FixDrinks(Drink drink1, Drink drink2) {
    long prepareEndTime;
    mental_act {prepareEndTime = System.currentTime() + randGen.range(3, 10);}
    with (priority_when_fails: subgoal PrepareDrinks(prepareEndTime);
        subgoal AnimEngine_CreateDrinks(drink1, drink2);
        subgoal PickupObjects(drink1, drink2);
        subgoal OfeObject(drink1, eObject_player);
    }
    sequential behavior PrepareDrinks(long prepareEndTime) {
        int whichAnim;
        mental_act {whichAnim = eAnim_fixDrinks1 + randGen.range(0, 9);}
        act DoAnimation(whichAnim);
        if (System.currentTime() < prepareEndTime) fail;
    }
}
```

**Example: Continuous monitoring**

```plaintext
sequential behavior FixDrinks(Drink drink1, Drink drink2) {
    context_condition (HasObjectWME characterID == eObject_player, objectType == eObjectType_barObject)
}
parallel behavior PrepareDrinks(long prepareEndTime) {
    subgoal PrepareDrinks_seq(long prepareEndTime);
    with (optional) subgoal PrepareDrinks_smileWhenPlayerClose();
}
sequential behavior PrepareDrinks_smileWhenPlayerClose() {
    with (success_test: ObjectPositionWME objectID == player, x > 50) wait;
    subgoal SetFacialExpressionBase(eFEBase_smile);
    subgoal DoGaze(eObject_player);
}
```

**Example: Low-level parallelism**

```plaintext
parallel behavior BeatGoal_IntroduceAnniv() {
    with (effect_only) subgoal BeatGoal_IntroduceAnniv_staging();
    subgoal BeatGoal_IntroduceAnniv_seq();
}
parallel behavior BeatGoal_IntroduceAnniv_staging() {
    with (priority_modifier 1) subgoal WalkTo(eObject_player);
    subgoal KeepFacing(eObject_player);
}
sequential behavior BeatGoal_IntroduceAnniv_seq() {
    subgoal SetFacialExpressionMood(eMood_happy, eMoodStrength_low);
    subgoal SetFacialExpressionBase(eFEBase_serious);
    subgoal DoGaze(eObject_player);
    subgoal DoDialog(etripScript_dialog_oh_huh_i_just_thought_of_something);
    subgoal SetFacialExpressionBase(eFEBase_smile);
    subgoal DoGaze(eObject_grace);
    subgoal WaitFor(2);
}
```
Example: High-level behavior mixing

```java
conflict
FixDrinks
BeatGoal_IntroduceAnniv_staging;

parallel behavior TheBeat()
with (priority 10) subgoal BeatGoal_FixDrinks();
with (priority 1) subgoal BeatGoal_IntroduceAnniv();
}
```

Joint goals and behaviors

Characters sometimes need to coordinate action

Some approaches
- Coordinate through sensing (but plan recog. hard)
- Explicitly communicate (but ad hoc)
- Build it into architecture (but not flexible)

Architecture coordinates author-specified joint action

Negotiation
Conflicting intentions

Problem: asynchronous agents enter conflicting states

Resolution: intentions are precedence ordered

Inconsistent subtree execution

Problem: continuing execution leads to ABT inconsistencies

Resolution: freeze subtree

Variably coupled agents

A tunable spectrum between one-mind and many-minds
Example: Coordinating dialog

The leader has these behaviors

```plaintext
conflict
BeatGoals_Intro
BeatGoals_Question;
```

parallel behavior BeatGoals() {
with (priority 2) subgoal BeatGoals_FixDrinks();
with (priority 1, persistent when_fails) subgoal BeatGoals_Intro();
with (priority 0, persistent when_fails) subgoal BeatGoals_Question();
}

sequential behavior BeatGoals_Intro() {
joint subgoal BeatGoals_IntroduceAnniv_line1();
joint subgoal BeatGoals_IntroduceAnniv_line2();
}

Grace and Trip each define their version of _line1 and _line2

```plaintext
joint parallel behavior BeatGoals_IntroduceAnniv_line1() { teammembers Grace Trip; ... }
```

Meta-behaviors

- Meta-behaviors manipulate the runtime state of other behaviors (e.g. succeed or fail steps).
- Ability to match on this runtime state just like it was part of the world (preconditions, context conditions, success tests)

Conversation = joint behaviors + handlers

```plaintext
sequential behavior handlerDA_DoReaction() {
precondition {
(ReactionWME type == eDAType_Disagreement)
theBeat = (ParallelBehaviorWME signature == "TheBeat")
}
subgoal AbortBeatGoalAndSuspendTheBeat();
with (priority 10, persistent when_fails) spawngoal BeatGoal_DisagreementReaction() at theBeat;
subgoal UnsuspendTheBeat();
}
```

```plaintext
sequential behavior BeatGoal_DisagreementReaction() {
joint subgoal BeatGoal_DisagreementReaction_line1();
joint subgoal BeatGoal_DisagreementReaction_line2();
}
```
Interaction = (Joint) behaviors + handlers

- Difficult to specify responsive sequential activity
  - Implicitly encode in ABT – conditions get complicated fast!
  - Flat behaviors with declarative state – redundant and error prone
- Instead: Joint behaviors + handlers (meta-behaviors)
  - Explicitly encode sequential activity in ABT
  - Modify future activity through dynamic ABT modification

ABL Conclusions

Behavioral coding vs. FSMs
- Behaviors support mixing (can be in more than one “state” at once)
- Behavior hierarchy more expressive than flat FSMs
- Dynamic coupling between goals and behaviors

Behavioral coding vs. rules
- Behaviors support sequential activity
- Behaviors support hierarchy