## 4.4.2 Environmental infrastructure

The environmental infrastructure consists of a clock system, a scenery management system, and an ambient sound control system. PAW^2 provides a sense of time and four distinct seasons using the environmental infrastructure to reduce the chance of users becoming bored or disinterested. The scenery of PAW^2, in-world ambient sounds, and content of shops in the world are changed over time. For example, users can see that cherry blossoms are in full bloom in PAW^2's spring season and maple leaves on the ground in its autumn season (see Figure 4.8). Also, users can hear the sound of birds singing during PAW^2's morning and cries of owls during PAW^2's night. These were implemented based upon typical natural phenomena in Japan.

PAW^2 has a different clock system compared with the real world. A single day (24 hour period) in PAW^2 is equivalent to two hours in the real world, and one season is equivalent to one week in the real world. So one year in PAW^2 is equivalent to four weeks in the real world.

We defined these configuration parameters based upon our assumption about user's behavior in PAW^2. The PAW^2 world has an area of 0.5 square km and consists of four islands (see Section 4.6). Each island has houses, buildings, trees, and roads. The total length of the road where a user can walk is about 7.8 km. The underlying CP system which builds up PAW^2 enables developers to specify the rate at which the user can travel through a world in meters per second. In PAW^2, it is specified as 1.0 m/s. It means that the 7.8km (7,800m) takes 7,800 seconds (about 2.1 hours) of walking time. Based on the result, we decided two hours as one day in

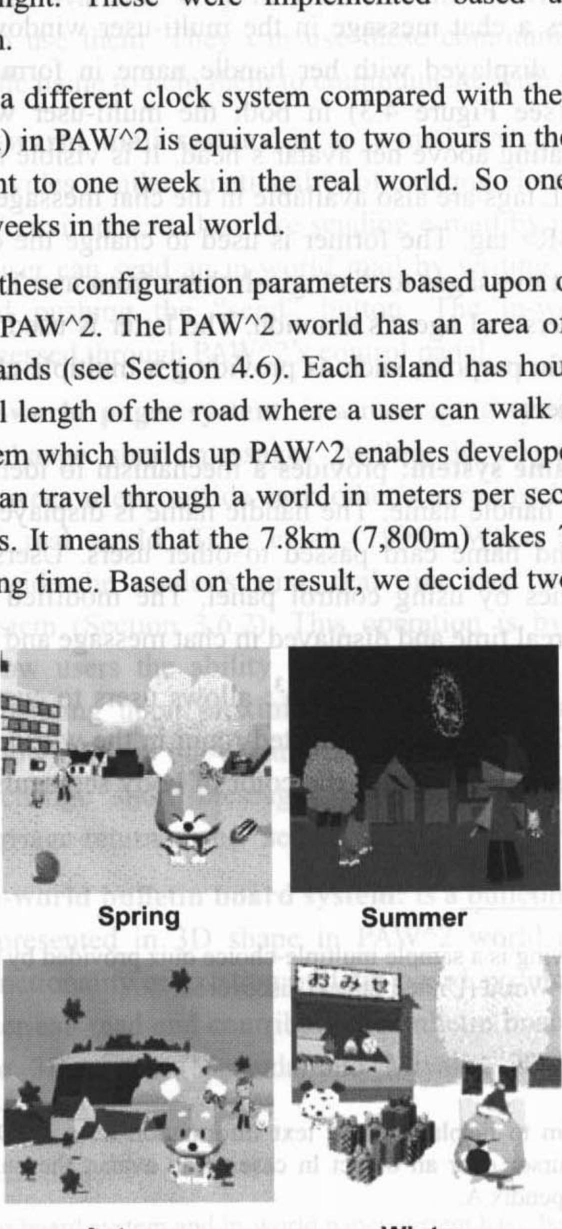

Figure 4.8 Four seasons in PAW^2

the PAW^2 world. In establishing the length of one season in PAW^2, we assumed that users can access the Internet at least once per week. We decided that one year in PAW^2 is equivalent to four weeks. This allows users to enjoy a different PAW^2 season upon each Internet access. For example, when they access PAW^2 every weekend, they can encounter a different PAW^2's season.

## 4.5. Events

As described in the previous section, we introduced the concept of time (sense of season and clock system) to give variation to the scene to reduce the chance of users becoming bored or disinterested. In addition to that, we hosted several events (mini games) periodically in PAW^2 to allow users to have various shared experiences. All users can freely participate in the events. These events consist of two types: periodic events and the events which are always available for users.

For example, as periodic events, we provided a different event for each of the four seasons. (1) During every PAW^2's spring season, a user can purchase flower seeds in a shop and plant them in the PAW^2 world. Some of the seeds produce flowers which appear in the following PAW^2's summer season (Figure 4.9). Then the user can pick the flowers and present them to her friend. (2) With the start of the PAW^2's summer, several "Omamori-Jizou" guardians appear in PAW^2 world. When a user clicks on one of them, it will fly her to a random place in PAW^2 world. The user has to walk back to the place where the item was and click on it again to get it. (3) In the PAW^2's autumn season, a lot of Japanese maple leaves appeared and users can click these to gather them. The leaves are tickets in a public lottery and users can win presents (special items) when the prize winners are drawn during the following PAW^2's winter season. (4) At the start of PAW^2's winter season, Santa Claus appears in PAW^2 world to ask users to deliver presents. Santa Claus hands out presents to users and requests delivery to houses with a Christmas

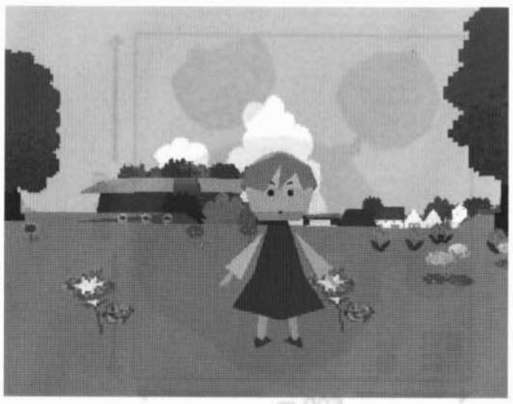
Figure 4.9 Flowers in the PAW^2's summer season.

stocking hanging on the doors of the houses. Participating users can obtain a special item for helping Santa Claus. We also host special events independent to the periodic events, e.g. an orienteering event based upon PAW^2's geography.

As the events which are always available for users, we hosted "Search for a cat!" (game) event, "Daruma stacking" (game), "Buru-Piko" game, and fortune teller. "Search for a cat!" is a tag game to find a special cat. To play the "Daruma stacking" game, teams consisting of three players compete to stack three "Daruma" tumblers vertically in the fastest time. "Buru-Piko" game is a single-user mole attack arcade game. Fortune teller is a game to get a fortune from an in-world oracle. "Search for a cat!" event, "Buru-Piko" game, and fortune teller are deigned for a single user. See Section 5.2.1 for detail about them. The results ranking of every event was announced on PAW^2's home page.

In some events, a personal agent plays an important role in guiding her owners, such as giving hints to her owner by using her natural language functionality.

## 4.6. Size of the World

We decided the size of PAW^2 world based upon "how many other users can one user see simultaneously?" Figure 4.10 shows the bird-eye's view of PAW^2 world. It consists of four islands. Suppose that the main island of PAW^2 (the biggest island in Figure 4.10) has a circle area with the radius of $r$ meters. Meanwhile, each user's aura has a circle area with the radius of 15 meters in PAW^2. Therefore, if there are 500 users (half number of one thousand users) in the island and we expect that each user has two other users in her aura, we can calculate the $r$ using the following expression:

$$(\pi r\hat{}2/\pi(15)\hat{}2)\times(2+1) < 500$$

$$r < 193 \text{ (m)}$$

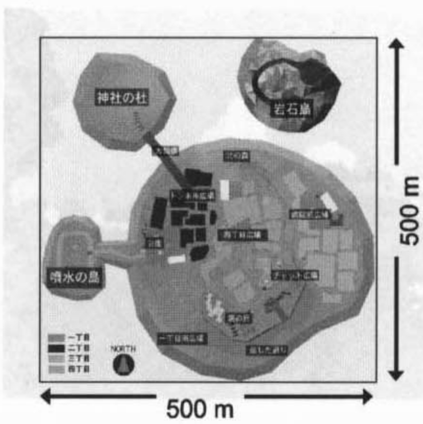

Figure 4.10 The bird's-eye view of PAW^2 world

In practice, the radius of the biggest island has 185 m and the world has an area of approximately 0.5 km by 0.5 km.

## 4.7. Architecture of PAW^2

Functions in a virtual society can be implemented in object-oriented architecture. Figure 4.11 shows an architecture of PAW^2 system. PAW^2 functions are implemented as a set of individual AOs and their managers. There are two types of managers: managers for their AOs and managers for other functionality in PAW^2 system. As we described in Section 3.6.1, consistency management is important in MUSVE. In PAW^2, the local consistency of each object is managed in each AO managing the object and its global consistency is managed in each manager for the AO. In the following sections, we describe these AOs and managers:

### 4.7.1 Universal AO

The Universal AO manages the entire PAW^2 world. It supports the following managers, Recycle Box AOs, and Shop AOs (Section 4.7.2):

- **User Manager:** manages users when they enter/exit the PAW^2 world. Each user is managed by a User Object in PAW^2 system. When a user enters the world, the User Manager authenticates her and creates a User

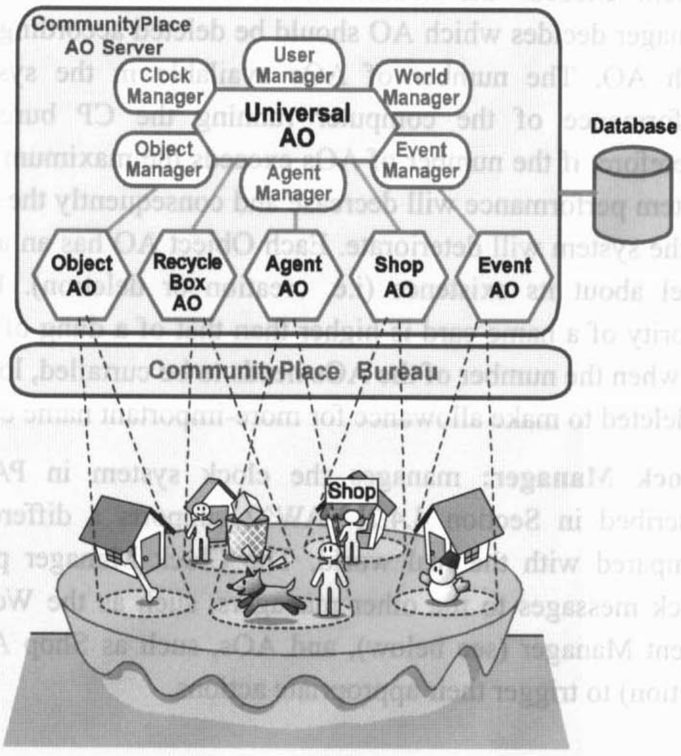

Figure 4.11 An architecture of PAW^2

Object for her. The User Object retrieves the user's information from a database to populate the information into the object. Then it sends a message to the Agent Manager (see below) to create her personal agent (i.e. Agent AO, see Section 4.7.2). When a user exits PAW^2, the User Object stores the user's latest information into a database and sends a delete message to the Agent Manager. Then, the User Object sends its delete message to the User Manager and quits itself.

- **Agent Manager:** manages creation and deletion of personal agents, i.e. Agent AO (Section 4.7.2). Basically, it receives create/delete messages about Agent AO from other managers or AOs to carry out its creation/deletion. In addition, it manages the number of Agent AOs in PAW^2 world to keep within its predefined fixed number. If a new user accesses PAW^2 (i.e. in case of receiving the create message from User Object), this manager checks whether her personal agent has been created or not. If the personal agent is not created, the manager deletes one of existing Agent AOs and then creates (or reuses) the user's Agent AO.
- **Object Manager:** manages creation and deletion of objects, such as items in PAW^2. The object is implemented as an Object AO (Section 4.7.2). This manager manages Object AO and its derived AOs. Basically, it receives create/delete messages about Object AO from other managers or AOs to carry out its creation/deletion. In addition, it manages the number of Object AOs in PAW^2 world. When the number of the AOs in system exceeds the maximum number of them in the system, this manager decides which AO should be deleted according to the priority of each AO. The number of AOs available in the system depends on performance of the computer running the CP bureau or the AOs. Therefore, if the number of AOs exceeds the maximum number, the total system performance will decrease and consequently the quality of service in the system will deteriorate. Each Object AO has an assessable priority level about its existence (i.e. creation or deletion). For example, the priority of a name card is higher than that of a dung of a personal agent. So when the number of the AOs needs to be curtailed, lower-priority dung is deleted to make allowance for more-important name cards.
- **Clock Manager:** manages the clock system in PAW^2 world. As described in Section 4.4.2, PAW^2 supports a different clock system compared with the real world. The Clock Manager periodically sends clock messages to the other managers, such as the World Manager and Event Manager (see below), and AOs, such as Shop AOs (see the next section) to trigger their appropriate actions.

- **World Manager:** manages the scenery and ambient sound in the PAW^2 world. This manager mainly supports the environmental features (see Section 4.4.2). It consists of a scenery management system and an ambient sound control system. When this manager receives a clock message from the Clock Manager, it changes the scenery and ambient sound according to the present season and time in PAW^2. It changes the color of sky and ground, appearance of trees, and the shape of clouds. For example, this manager allows users to see trees with cherry blossoms when PAW^2's spring has come.
- **Event Manager:** manages events (see Section 4.5) hosted in PAW^2. It reads a configuration file for events and carries out the predefined events for specified season and time in PAW^2. We also provide a middleware to allow ISVs (Independent Software Vendor) to easily develop their original events (Matsuda and Miyake, 2000).

When the PAW^2 system starts, the Universal AO initializes each manager and then requests the Agent Manager to create the predefined number of personal agents, shops, and recycles boxes at pre-defined positions by default.

## 4.7.2 Individual AOs

Individual shared applications, such as a personal agent, provided by PAW^2 are implemented as an AO. In the PAW^2 system, there are several types of AOs for their tasks.

- **Agent AO:** manages a personal agent. This AO has functions to walk around in the PAW^2 world autonomously, process natural language chat messages and user's mouse operation, control her internal states, assist her owner, and take appropriate actions to her owner, AOs, and other users. For example, a personal agent offers a greeting to her owner's friends when the agent meets them, and can eat food given to the agent (Section 4.7.6). In addition, this AO has a function to send e-mail to her owner. Depending upon the internal state, this AO changes the appearance or actions of a personal agent. When a user enters PAW^2 world, the Universal AO requests the Agent Manager to create an Agent AO and assigns it to the user (Section 4.7.6). See also Section 4.7.3 about architecture of Agent AO.
- **Object AO:** manages an "object" (i.e. item) existing in PAW^2 world, including food, dung of personal agents, flowers, seeds, name cards, accessories, and so on. This AO provides a base class of these objects. It is easy to extend to realize various types of objects, such as objects that

contain information (e.g. a name card), objects that affect the internal state of personal agents (e.g. food for a personal agent), or objects that are deleted at a predefined time (e.g. a personal agent's dung). When a user places an object into the PAW^2 world, the Universal AO receives a "put" message from the user and requests the Object Manager to create an Object AO for the object (see Section 4.7.6).

- **Shop AO:** manages a shop which sells goods (items), such as food for personal agents, accessories, or special items for an event. This AO has capability to display goods (items) for sale. When a user clicks a sale item, this AO communicates with her User Object to sell the item. Thereafter, this AO takes virtual money according to the price of the item from her pocket (database) and then stores the item into the database. When this AO receives a clock message from the Clock Manager, it changes content of the shop according to the current season and time.
- **Recycle Box AO:** manages a recycle box which allows users to recycle garbage objects into virtual money i.e. POLYGO. This AO is an implementation based upon the Shop AO. In case of a shop, the AO exchanges virtual money for an object. In case of a recycle box, the AO exchanges an object for virtual money.
- **BBS AO:** manages an in-world BBS system (see Section 4.4.1). This AO provides a 3D user interface of BBS and access methods to the BBS database (Matsuda and Miyake, 2000). When a user accesses its user interface, this AO retrieves articles from the database to allow her to read them. When a user writes a new article to the BBS, this AO stores it into the database.
- **Event AO:** manages events. This AO is generated by the Event Manager according to the event schedule. The Event Manager reads a configuration file for events and then generates the appropriate Event AOs (Matsuda and Miyake, 2000).

These AOs are exchanging messages, thus are interacting with each other. These AOs also communicate with a database to save their internal states to allow PAW^2 system to recover all the previous states if the system goes down.

## 4.7.3 Agent AO

A personal agent can react to several events: user's chat messages, mouse operations, periodic timer, user's login/logout, encounters with users/AOs, and PAW^2 events. To deal with these events, Agent AO consists of four components: Event Dispatcher, AgentAction, AutoTalk, and State Manager (Figure 4.12).

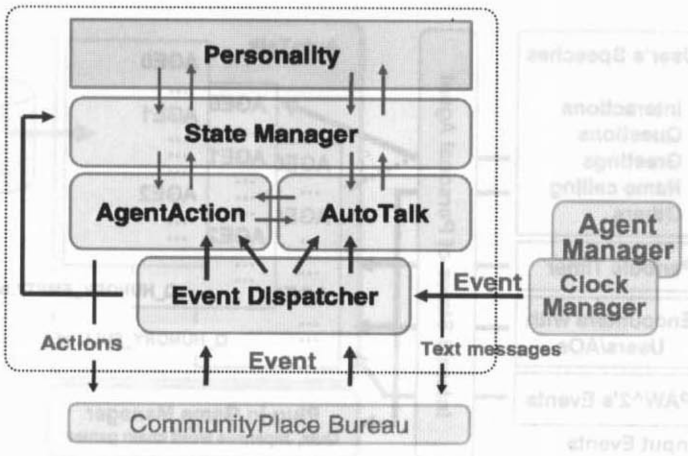

Figure 4.12 An architecture of Agent AO

## Relationship among components

When the Event Dispatcher receives events from CP bureau or Clock/Agent Manager (Section 4.7.1), it decides the target module to dispatch the events based upon their meaning. Simultaneously, it converts the (external) event into an internal event. The AgentAction module manages the actions of a personal agent and decides which actions should be carried out for the events. The AutoTalk module manages the speech acts of a personal agent and reacts to the events by using natural language. We will describe this module later. The State Manager manages the internal states of a personal agent such as growth and emotion. The "periodic timer" event allows the manager to change the internal states periodically and store them in a database. The internal states affect the appearance or personality of a personal agent.

These modules work and interact with each other. For example, when a user strokes her personal agent by using a mouse, the event is passed to the AgentAction module to carry out her delight actions. Then the module requests the AutoTalk module to say something like "I am happy." When a periodic timer event is passed to the State Manager and the personal agent becomes hungry, the State Manager requests the AutoTalk module to say something like "I am hungry." Then, the module requests the AgentAction module to carry out a series of actions representing her hungriness.

## Conversation module

As described in Section 4.3.3, a personal agent has a natural language processing function extended on ELIZA (Weizenbaum, 1966). Figure 4.13 shows the conversation mechanism called "AutoTalk" in a personal agent.

When the AutoTalk module receives an event from the Event Dispatcher, the module selects one text message from a database based upon both the event and the

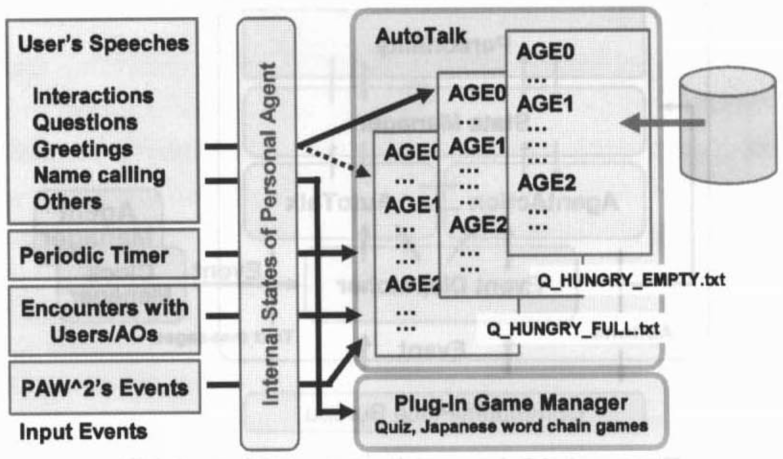

Figure 4.13 The mechanism of AutoTalk

internal states of a personal agent. Then it requests the personal agent to speak out the message. If necessary, the AutoTalk also requests AgentAction module to take an appropriate action for the message (see Figure 4.12). Some of user's chat messages such as "Let's play a quiz game!" are directly passed to the Plug-in Game Manager to start the game. The Plug-in Game Manager manages a set of text-based games (e.g. quiz or Japanese word chain games). It carries out one of the games corresponding to the inputted chat message. It also provides a middleware to allow application developers to develop new text-based games which can be plugged-in a personal agent.

Meanwhile, the periodic timer event realizes autonomous chat from a personal agent. The "Encounters with Users/AOs" event allows a personal agent to offer a greeting to other users or agents. It also stops her autonomous chat in case more than four users are in her aura. PAW^2's events (Section 4.5) also could carry out the AutoTalk module, for example, to give their hints or rules.

The extended ELIZA system in PAW^2 provides 9,558 patterns. Text messages of a personal agent are stored into the text files based upon the following information: types of input events and internal states of a personal agent. For example, in Figure 4.13, Q_HUNGRY_FULL.txt contains text messages that a personal agent speaks out when her owner asks "Are you hungry?" and she is full. Each text file contains a personal agent's age and corresponding text messages for the age. The AutoTalk module needs to select a text message from those corresponding to the current age of a personal agent.

### 4.7.4 Geography manager

The Geography Manager manages geographical information describing a virtual world in order for AOs to retrieve the information in a low-cost manner. PAW^2's geographical information is stored in a resource file called "map.dat" located at the server side system. This file contains the geographical information in a linked-list format (Figure 4.14) for describing paths or areas in a virtual world. Each link describes a path (or an area), its width, and connections between two nodes. Each node has its coordinate in a virtual world. This manager provides several convenient methods to retrieve the information, i.e. coordinates, vector, and width between two specified nodes. AOs or managers can send a message to the Geography Manager to retrieve the information.

For example, when a user places an object into PAW^2, the Object Manager needs to locate the object in the virtual world. In this case, the Object AO communicates with the Geography Manager to decide its position to be placed in the world. In particular, a personal agent needs to walk around in the world autonomously towards a programmed destination. In this case, the personal agent needs to walk to the destination without colliding with other agents or obstacles, such as houses in the virtual world. The computational cost of these calculations is high in a world inhabited with a lot of personal agents or obstacles like PAW^2. The Geography Manager can reduce this calculation cost, especially the cost of collision detection among shared applications.

## 4.7.5 Other components

Besides the components described above, there are several other software modules to realize PAW^2's functions. The In-world Mail Manager manages e-mail communication carried out using in-world mailers. This manager receives in-world mails from a sender User Object and stores these in a database. When this manager receives a "read" request from a user (User Object) who reads the mails, it retrieves her mails from the database and sends them to her. The In-world Pager Manager manages short text communication carried out using in-world pagers. This manager receives a text message from in-world pager of the sender User Object and sends it to the target User Object directly. The In-world BBS Manager manages the In-world BBS system. The Database Manager controls access interface to a database from AOs and managers. The Internet Mail middleware allows AOs and other PAW^2 system components to send an Internet mail to users.

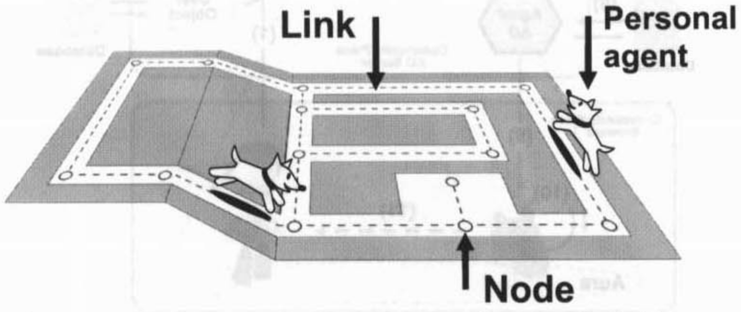

Figure 4.14 Geographical information for PAW^2