D3PART: A new Model for Redistribution and Plasticity of 3D User Interfaces

Jérémy Lacoche, Thierry Duval, Bruno Arnaldi, Éric Maisel, Jérôme Royan

D3PART:
A new Model for Redistribution and Plasticity of 3D User Interfaces

Jérémy Lacoche*
IRT b<>com
UMR CNRS 6074 Irisa - Inria Rennes

Thierry Duval†
UMR CNRS 6285 Lab-STICC
Telecom Bretagne
IRT b<>com

Bruno Arnaldi‡
UMR CNRS 6074 Irisa - Inria Rennes
INSA de Rennes
IRT b<>com

Eric Maisel§
UMR CNRS 6285 Lab-STICC
ENIB
IRT b<>com

Jérôme Royan¶
IRT b<>com

ABSTRACT

In this paper we propose D3PART (Dynamic 3D Plastic And Redistribuable Technology), a model to handle redistribution for 3D user interfaces. Redistribution consists in changing the components distribution of an interactive system across different dimensions such as platform, display and user. We extend previous plasticity models with redistribution capabilities, which lets developers create applications where 3D content and interaction tasks can be automatically redistributed across the different dimensions at runtime.

Keywords: Plasticity, Redistribution, 3D User Interfaces

Index Terms: H.5.1 [Information interfaces and presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; H.5.2 [Information interfaces and presentation]: User Interfaces—Graphical user interfaces (GUI)

1 INTRODUCTION AND RELATED WORK

Today, users have access to a wide variety of platforms such as mobile devices, desktop computers and immersive systems. Therefore, users are more frequently confronted with situations where they have to move from one platform to another [7]. Moreover, combining different platforms can give new interactions prospects to users. These possibilities directly refer to "distributed user interfaces" (DUI) and redistribution. A DUI is a user interface whose components are distributed across different dimensions [8]. For 3D user interfaces we consider three dimensions of distribution from the ones described in [8] and [15]:

- **Display**. The application content is displayed on one or multiple devices. Common examples in 3D for this kind of distribution are multiple display systems.
- **Platforms**. The application runs on a single computing platform or is distributed across multiple ones. These platforms may be homogeneous or heterogeneous (operating system, computing power, plugged devices). For instance, cluster approaches combine connected homogeneous computers to run a VR application with high performances.
- **Users**. The application is shared by multiple users. This dimension is directly linked to the two other ones as the different participants can use different displays and platforms. In 3D, this dimension refers to Collaborative Virtual Environments (CVE).

Redistribution consists in changing the distribution of an interactive system on these dimensions. It can be system-initiated, user-initiated, or mixed-initiated [7]. Redistribution can be performed at runtime or between sessions and its granularity may vary from application to pixel level [4]:

- At **application level**, on the platform or user dimension, the application is fully replicated or fully migrated on a distant platform. The application may be adapted to its new context of use. Full replication implies state synchronization to maintain consistency between the different instances of the application , while for a full migration no synchronization is performed.
- At **workspace level**, workspaces can be redistributed on the three dimensions. A workspace is an interaction space that groups together interactors that support the execution of a set of logically connected tasks. For instance, the painter metaphor [16] includes two workspaces: the palettes of tools on a mobile device and the drawing area on an electronic white board.
- At **domain concept level**, physical interactors can be redistributed on the different dimensions. In 3D, it corresponds to the interaction techniques and widgets. In [14], physical interactors for navigation, pointing and application control are distributed on a tablet in order to interact in an immersive system.
- At **pixel level**, view continuity is ensured across different displays with a distribution on the display and the platform dimensions. For instance, an application can be distributed on a cluster of PCs and rendered on multiple displays with view continuity.

Redistribution is one mean of adaptation addressed by plasticity which is the capacity of an interactive system to withstand variations of both the system physical characteristics and the environment while preserving its usability [18]. The second mean of adaptation addressed with plasticity is recasting which consists in modifying locally the application components in order to fit a given context of use, such as interaction techniques adaptations or content presentation modifications. Recasting is needed to handle redistribution, because input and output capacities variations from a platform to another one imply local adaptations of the redistributed components. In 3D, solutions exist for the creation of reconfigurable applications [9], adaptive ones [13] and recent approaches tend to bring plasticity to 3D with a focus on recasting [12].

In order to handle redistribution, most of the solutions are designed for 2D user interfaces, such as the 4C reference framework

*e-mail: jeremy.lacoche@b-com.com
†thierry.duval@telecom-bretagne.eu
‡e-mail: bruno.arnaldi@irisa.com
§maisel@enib.fr
¶jerome.royan@b-com.com

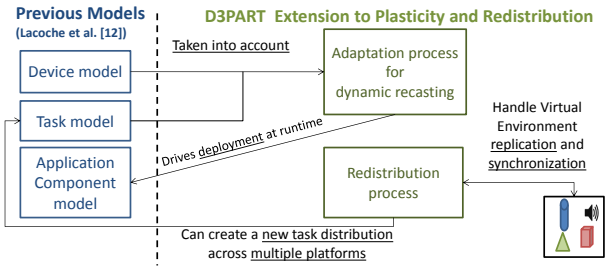

Figure 1: D3PART extends our previous plasticity models [12] by integrating an adaptation and a redistribution process.

[7], the peer-to-peer architecture proposed by Melchior et al. [15], the PolyChrome framework [2], or the ZOIL framework [19]. Solutions to create distributed 3D user interfaces also exist but they mainly focus on specific cases and do not let the end-user change the system distribution at runtime. One specific case handled in 3D is the case of clusters of computers that manage multiple display systems such as CAVEs [6] or Workbenches. VR Juggler [3] and MiddleVR[1] propose such solutions. The second specific case handled in 3D is the field of CVE which needs a distribution at the platform and user levels. It implies a state synchronization between the different users platforms in order to maintain a consistent application. Some architectures for CVE are reported in [10].

Our contribution is D3PART (Dynamic 3D Plastic And Redistribuable Technology), a new model for developers to help them in the creation of 3D user interfaces that can be dynamically redistributed across different dimensions: platform, user and display. The model includes an adaptation process and a redistribution process for the creation of plastic 3D applications. We focus on redistribution at application, workspace, and domain concept levels. Pixel level on clusters of PCs is not covered. We present one scenario of redistribution where we combine a tablet and an immersive system for a furniture planning application. This prototype is developed with a toolkit that implements the D3PART model.

2 Application Model and Dynamic Recasting

As shown in Figure 1, in order to design 3D applications that handle plasticity, recasting and redistribution, D3PART extends our previous plasticity models [12]. First, this previous work introduces a device model for the description of any platform. This device model describes precisely all the devices that can be used for interaction purposes at runtime. It includes device capabilities, limitations and representations in the real world. Second, it introduces a model for developing concrete application components independently from any 3D framework or 3D devices. These components are deployed at runtime to achieve high level interaction tasks which are also represented in a model. For 3D user interfaces, according to Hand [11], these tasks belong to three categories: selection and manipulation, application control, and navigation. For instance, an application component can correspond to an interaction technique or a 3D widget. This model is a modification of PAC [5] and ARCH [1] models. It divides a component into five facets that decouple its features:

- The Abstraction describes the semantics of the component and the function it can perform.
- The Rendering Presentation facet is the only facet depending on a 3D framework. It handles graphics output and physics. In our examples these facets are developed with Unity3D[2].
- The Logical Driver handles devices management. It can implement how an interaction technique is controlled according to a set of abstract interaction devices. In this facet, the developer describes all required inputs and outputs units according to a set of parameters taken from the device model.

[1] http://www.middlevr.com/middlevr-sdk/
[2] https://www.unity3d.com/

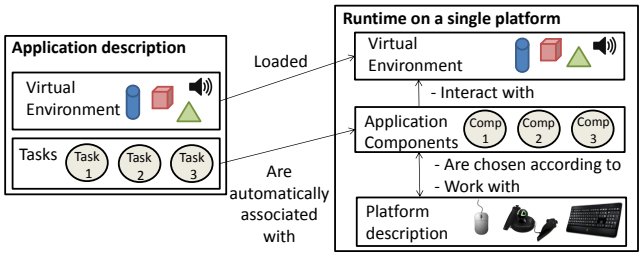

Figure 2: With D3PART, an application is described by a virtual environment and high level tasks. Compatible applications components are deployed to achieve the tasks according to the encountered context of use.

- The Control ensures the consistency between the rendering presentation, the logical driver and the abstraction.
- The Supervision Control receives the context modifications at runtime and then is able to determine if a logical driver is still possible. It contains all logical driver and rendering presentation types compatible with the application component.

D3PART uses these models to describe an application and the context of use. As shown in Figure 2 we define an application with a set of high level interaction tasks and with a description of the virtual environment. First, the application developer chooses a set of tasks to represent at a high level the application behavior and possibilities. Dependencies between the tasks can be described by the developer. For instance, an application control task with a menu will be dependent to a selection task. In our implementation, these needed tasks and the dependencies must be provided by the application developer or the designer in an XML configuration file. New tasks can be implemented by a developer and added to the list of possible ones. A task can define different functions (the task events) that constitute the application logic such as adding an object into the scene or loading a new scene configuration, etc. A task also exposes a list of compatible application components that can be deployed to achieve it. This list is also edited in an XML file. These components must be implemented with the application component model described in [12]. This previous work gives examples of possible tasks, application components and logical drivers.

Second, the application is described with its virtual environment. The virtual environment is composed of visual (3D content) and sound assets. Its edition is separated from the tasks. It can be edited separately, for instance in a game engine editor, or loaded with an X3D file depending on the implementation of the models used. In our case, as said, we use an implementation based on Unity3D.

The application is launched on a platform described with the device model previously introduced. Each device corresponds to a class that inherits from the basic device class. In this class, with the device SDK, the developer has to complete some functions to fulfill the input data, trigger the outputs and tell the system when a new instance of the device is plugged or unplugged.

At runtime, high level tasks are automatically associated with concrete application components according to the encountered context of use. For these components, the rendering presentation facet and the logical driver facet are also chosen according to the context. The control facet and the abstraction one do not depend to this context. The association is performed with an automatic adaptation process included in D3PART on top of the device and task models in order to support dynamic recasting. The association is made with a scoring system that takes into account the platform capabilities and the list of compatible components exposed by each task. Its goal is to maximize the usability of the application. We won't give a full description of this scoring mechanism because it is not the scope of this paper. The association process is performed at each context change in order to detect not any longer usable application components or more adapted ones. It can be described as follows:

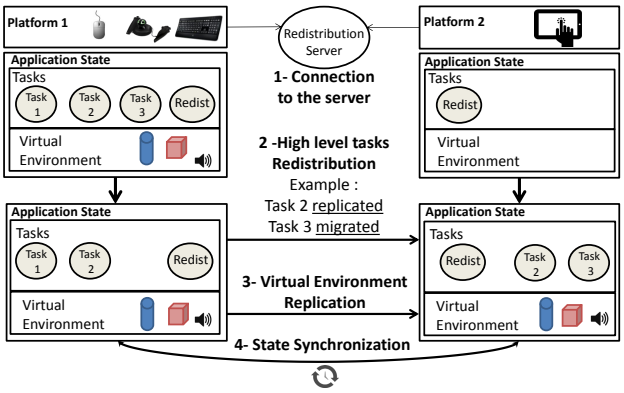

Figure 3: The four steps of D3PART redistribution process.

1. A context modification is detected. For example, it can be the connection of a new device or the add of a task.
2. For each deployed application component, we check if the associated logical driver is still possible in the current context of use. It is still possible if the devices that it uses are still plugged and available. If not, the application component is destroyed and the associated task is classified as not done.
3. For each not done task, we create a list of all possible triplets (application component, logical driver, rendering presentation) that can achieve it. A triplet is can be instantiated if the logical driver needed device units can be found in the list of available devices. Only the rendering presentations in the current 3D framework can be used. A compatibility score is attributed to each triplet. The one with the best score is deployed. The devices units associated with the logical driver are set as not available. The task is classified as done.
4. For each done task that has not been processed in the previous step, we check if we can find a triplet more adapted than the current one. This optimization is not performed at the same time than the previous step. Indeed, the priority is given to the association of application components to the not done tasks. For this optimization, we check if we can find a triplet with a better score than the current one. If we find one, we destroy the current component and we deploy the new best choice.

With this adaptation process, dynamic recasting is supported and optimal usability of the application is always ensured whatever the context of use. It will allow the application to handle the different context changes encountered during the redistribution process.

3 Redistribution Process

As shown in Figure 1 D3PART includes a redistribution process that makes the integration of redistribution capacities totally transparent and automatic for the developer. The process consists of distributing the high level tasks and the virtual environment across the different dimensions: platform, display and user. The developer's work is to create high level tasks and implement the compatible application components with the help of the models from [12] previously described. With the implementation of multiple compatible components for each task and multiple logical drivers, that use different kinds of devices, for each component, the developer ensures that his application will be usable on a wide variety of platforms.

We added a built-in high level task and its corresponding application component in order to allow any developer to add redistribution capability to his application. For this component, the abstraction facet contains the redistribution logic and the rendering presentation facet contains the parts that are dependent to the target 3D framework. The redistribution process needs a connection mechanism between the different platforms for state synchronization and platforms discovery. To do so, we use a client/server architecture where the different platforms can register. For now, this feature is implemented with the network capabilities of the target 3D framework. It is integrated into the rendering presentation facet. As future work, this mechanism could become independent of the 3D framework and be implemented in the abstraction facet. For now, our implementation does not show apparent latency but being independent from the 3D framework would let us optimize the network load. As proposed in the 4C reference framework [7], this component implements an integrated user interface for platform registration and control redistribution process: the meta-user interface. In our case, the redistribution is performed at runtime and is user-initiated: the meta-user interface is proposed to the end-user of the application. It can be shown and hidden at runtime with a graphical button or a device button depending on the context of use. The redistribution process is performed in four steps as shown in Figure 3.

The first step consists in connecting to the redistribution server. The IP address of the server can be given in the meta-user interface or in the XML task configuration file. This step must be performed on the current used platform and on each platform that must be available for redistribution. On the distant platforms, an empty application runs. It contains the framework that implements the D3PART model and it declares the redistribution task as needed.

The second step consists in configuring the desired redistribution with the meta-user interface. First, the user chooses the platform on which the application will be redistributed from a list of available ones. In our case, the basis of the redistribution process is made on the platform dimension. However, as each platform may manage another display and may be used by another person, user and display dimensions can also be targeted. Then, the user configures the high level tasks distribution across the two platforms. As shown in Figure 4, multiple choices are given to the user in the menu:

- Full migration: all tasks migrate. Each platform runs an independent version of the application. It can be performed when the user wants to switch to another platform.
- Partial migration: the user chooses which task(s) will migrate to the distant platform. The application is distributed and so shared between the two platforms. It can be performed to combine different platforms.
- Partial replication: the user replicates some tasks to the distant platform. He will be able to perform these tasks on the two platforms within the same shared application.
- Full replication: all tasks are replicated and can be performed on different platforms in the same shared application. This kind of redistribution can be used to start a collaboration with a user on a different platform.

Dependent tasks have to be redistributed together. Therefore, they are grouped into the menu as shown in Figure 4. In the meta-user interface we associate a warning icon to a task if it cannot be performed on the distant platform. To do so, we ask the distant platform if an application component can be deployed for each task according to the platform capabilities. The goal of this feature is to warn the end user that the application can be degraded if this task is redistributed. On the other platform, thanks to adaptation process included in D3PART, an adapted application component is automatically associated with each redistributed task.

In the third step we replicate the virtual environment to the distant platform. The goal is to keep the application state during the redistribution to the target platform. It includes 3D meshes, their materials, and sound assets. To do so, we consider three solutions:

- Assets are known in the distant platform. Only the names are transmitted. This is the currently implemented solution.
- Assets are not known but can be downloaded from a distant server. In this case, URLs are provided.
- Assets are unknown. For instance when a user is editing a new 3D content. Here, assets can be streamed over the network.

Figure 4: The meta-user interface enables the user to choose the new task distribution.

In the last step we synchronize the different platforms. As for CVEs, a synchronization is performed in order to keep a consistent state between the instances of the same application running on different platforms. In case of full migration, no synchronization is performed because each platform runs an independent version. First, the 3D objects transforms are synchronized in order to maintain a consistency between the different 3D worlds. Second, tasks events are also synchronized. The events constitute the application logic and have to be synchronously performed on each application instance. To do so, we use an observer design pattern. The redistribution component observes all task events. When one event is triggered, it is transmitted with its parameters through the network as text messages in order to be triggered distantly. During a full replication, a collaborative context of use can be created. To handle concurrency when moving objects, the priority to move an object is given to the first user who grabs it. Other users cannot move an object until the first user has released it. Other mechanisms could be integrated as well. We also provide awareness about the activity of the distant user, for now we only display the view frustums of each user but avatars and hands could be added too.

4 Redistribution for Platforms Combination

The implementation of the D3PART model have been used to develop a furniture planning application. Its goal is to help people to plan the use of particular premises. Here, we demonstrate how two different platforms can be combined to interact in this application thanks to the D3PART model. The application is composed of three tasks. First, a navigation task is needed in order to navigate within the room. Second, we need an application control task for adding furniture into the room with the help of a menu. The add of an object is defined as an event into the task. Last, we need a selection and manipulation task for moving furniture and for menu selections. These two last tasks are defined as dependent: indeed selection possibilities are needed for interacting with the menu.

In this scenario we use an Android tablet and a CAVE with active stereo. MiddleVR is used to handle the different screens and clustering. Some novice users may not be confident with 3D interactions and may prefer more commons multi-touch interactions. With D3PART, the user can distribute the selection and manipulation operations on the tablet and the navigation in the CAVE. The user will be able to interact with the usual and easy-to-use tablet multi-touch capabilities while being immersed at scale one in the CAVE. The tablet would act like a remote World-In-Miniature [17]. To do so, the user chooses a partial migration to the CAVE, only the navigation task migrates to the distant platform. Other tasks remain on the tablet. This choice is made with the meta-user-interface as shown in Figure 4. On the tablet, for the furniture control task, a 2D menu is instantiated with the list of furniture that can be added. For the manipulation task, an interaction technique based on the multi-touch capabilities of the tablet is deployed. With this technique the user can translate the objects onto the floor with one finger and rotate them around the up axis with two fingers. In the CAVE, an interaction technique based on a walking metaphor controlled with

Figure 5: A combination of a CAVE and a tablet with D3PART.

head tracking and a joystick is deployed for the navigation task. It places the point of view inside the room in order to immerse the user in it. At this time the application is distributed on two platforms and displays as shown in Figure 5. A remote World-In-Miniature is on the tablet and at the same time the user is immersed at scale one into the room in the CAVE. The synchronization of the 6 DoF transforms of the objects between the two platforms ensures consistency when the user moves an object on the tablet. As well, the command for adding an object into the room is also synchronized. Both systems runs approximately at 25 fps. The difference of frame rates does not impact the synchronization. The meta-user interface is also available in the CAVE. Therefore, the user can migrate back the full application to the tablet when he has finished.

Other scenarios of redistribution could also be imagined with D3PART. For instance, a user interacting on a tablet that migrates all his application to a CAVE in order to continue his work with 3D interactions. Another possibility is to fully replicate its application to a colleague platform in order to start a collaboration with him.

5 Conclusion and Future Work

D3PART is a new model to handle plasticity and redistribution for 3D user interfaces. With D3PART, redistribution can be performed on the display, platform and user dimensions and can target three levels of granularity: application, workspace, and domain concept levels. Redistribution can be performed at runtime by the user with an integrated user interface: the meta-user interface. Dynamic recasting handled by D3PART, with the included adaptation process, ensures usability continuity whatever the new distribution chosen.

Future work will consist in automating the redistribution process to make it possibly system-initiated or mixed-initiated, which could consist in finding the right platform or the right user for each task according to the platforms capabilities and the user preferences. We could also consider level of details during the virtual environment replication as each platform may not have all the same computation capabilities. Last, we will evaluate the system to assess its interest, its usability and its acceptability for end users.

References

[1] A Metamodel for the Runtime Architecture of an Interactive System: The UIMS Tool Developers Workshop. *SIGCHI Bull.*, 24(1), 1992.

[2] S. K. Badam and N. Elmqvist. Polychrome: A cross-device framework for collaborative web visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 109–118, New York, NY, USA, 2014. ACM.

[3] A. Bierbaum, P. Hartling, P. Morillo, and C. Cruz-Neira. Implementing Immersive Clustering with VR Juggler. In *ICCSA 2005*, pages 1119–1128, Berlin, Heidelberg. Springer-Verlag.

[4] G. Calvary, J. Coutaz, D. B. Thevenin, L., M. Florins, Q. Limbourg, N. Souchon, J. Vanderdonckt, L. Marucci, F. Paterno, and C. Santoro. The CAMELEON Reference Framework. *Deliverable D1.1*, 2002.

[5] J. Coutaz. PAC, on object oriented model for dialog design. In *Interact'87*, 1987. 6 pages.

[6] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: Audio Visual Experience Automatic Virtual Environment. *Commun. ACM*, 35(6):64–72, June 1992.

[7] A. Demeure, J.-S. Sottet, G. Calvary, J. Coutaz, V. Ganneau, and J. Vanderdonckt. The 4C Reference Model for Distributed User Interfaces. In *ICAS 2008*, pages 61–69, March.

[8] N. Elmqvist. Distributed user interfaces: State of the art. In *Distributed User Interfaces*, pages 1–12. Springer, 2011.

[9] P. Figueroa, M. Green, and H. J. Hoover. InTml: A description language for VR applications. In Web3D 2002, page 5358. ACM.

[10] C. Fleury, T. Duval, and V. Gouranton. Architectures and Mechanisms to Maintain efficiently Consistency in Collaborative Virtual Environments. In *SEARIS 2010*.

[11] C. Hand. A survey of 3D interaction techniques. In *Computer graphics forum*, volume 16, pages 269–281, 1997.

[12] J. Lacoche, T. Duval, B. Arnaldi, E. Maisel, and J. Royan. Plasticity for 3D User Interfaces: new Models for Devices and Interaction Techniques. In *EICS 2015*. ACM.

[13] I. Lindt. *Adaptive 3D-User-Interfaces*. PhD thesis, 2009.

[14] D. Medeiros, F. Carvalho, L. Teixeira, P. Braz, A. Raposo, and I. Santos. Proposal and evaluation of a tablet-based tool for 3D virtual environments. *SBC*, 4(2):31, 2013.

[15] J. Melchior, D. Grolaux, J. Vanderdonckt, and P. Van Roy. A toolkit for peer-to-peer distributed user interfaces: concepts, implementation, and applications. In *EICS 2009*, pages 69–78. ACM.

[16] J. Rekimoto. Pick-and-drop: A Direct Manipulation Technique for Multiple Computer Environments. In UIST 1997, pages 31–39. ACM.

[17] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM: interactive worlds in miniature. In *CHI 1995*, pages 265–272. ACM.

[18] D. Thevenin and J. Coutaz. Plasticity of user interfaces: Framework and research agenda. In *Proceedings of INTERACT*, volume 99, page 110117, 1999.

[19] M. Zöllner, H.-C. Jetter, and H. Reiterer. *ZOIL: A design paradigm and software framework for post-WIMP distributed user interfaces*. Springer, 2011.