Computational Visual Media
DOI 10.1007/s41095-xxx-xxxx-x Vol. x, No. x, month year, xx–xx

Article Type (Research)

# Resolving display shape dependence issues on tabletops

James McNaughton[1] ✉, Tom Crick[2], and Shamus Smith[3]

**Abstract** Advances in display technologies are transforming the capabilities – and potential applications – of system interfaces. Previously, the overwhelming majority of systems have utilised rectangular displays; this may soon change with the ubiquitous and pervasive nature of digital devices that facilitate human interaction. At present, software is usually designed assuming it will be used with a display of a specific shape; however, there is an emerging demand for tabletop-orientated systems to be capable of handling a wide range of potential display shapes. In this paper, the design of software for use on a range of differently shaped tabletop displays is considered, proposing a novel but extensible technique that can be used to minimise the influence of the issues of using different display shapes. Furthermore, we present observations on an implementation of the technique to explore how it can be used to adapt the layout of tabletop software to different display shapes, identifying a range of potential applications and research priorities.

**Keywords** Visual content management, irregular displays, screen design, multi-touch surfaces, tabletop displays, ubiquitous computing.

1 School of Education, Durham University, Durham DH1 3LE, UK. E-mail: j.a.mcnaughton@durham.ac.uk
2 Department of Computer Science, Swansea University, Swansea SA2 8PP, UK. E-mail: thomas.crick@swansea.ac.uk
3 School of Electrical Engineering & Computing, University of Newcastle, Callaghan NSW 2308, Australia. E-mail: shamus.smith@newcastle.edu.au

Manuscript received: 2018-03-08; accepted: 2018-xx-xx.

## 1 Introduction

The majority of tabletop software systems that provide visual feedback to a user are normally designed to do so with a particular display shape, with the most common of these shapes being rectangular. However, adapting displays to different shapes has always been possible through covering regions of a display [6]. This, in addition to new technologies such as circular liquid crystal displays [3, 7], allows the potential for different display shapes to be used in tabletop systems. Developers thus need to consider how to make their software agile enough to adapt to not only displays of varying sizes and aspect ratios, but also to displays of varying shape. The work presented in this paper explores a novel technique that would allow developers to adapt tabletop software to different display shapes.

The remainder of this paper is structured as follows: Section 2 outlines the current development and drive for non-traditional display shapes with tabletop systems; Section 3 identifies the issues arising from the use of different display shapes on a system; A technique to minimise the influence of the identified issues is proposed in the Sec. 4 section; in Section 5 we present the development of the technique into a software framework. Section 6 discusses the implications that arise from the technique's use, with future developments and our conclusions presented in Section 7.

## 2 Background

Traditional displays used in digital systems are overwhelmingly rectangular. However, an increasing number of non-rectangular displays are becoming available, for a variety of real-world applications. One such instance of a non-rectangular display is Toshiba's circular thin film transistor liquid crystal display [3]; this display is typically employed in vehicles to show dashboard and operating information. The display is designed to look similar to a typical dial on a car dashboard, thus requiring the display to be circular. The Motorola Aura [7] is a mobile phone that showcases another example of a non-rectangular (circular) display.

A non-rectangular display is utilised in the PyMT – A Multitouch Framework for Python [10] project. A circular display is made by projecting the system's output onto a circular surface. The '*Puddle of Life*' application is designed to tailor its visual output to this circular display shape; meaning that the software is constrained by the specifications of the hardware. This is also true for software built for circular dashboard [3] and phone [7] displays. As a result, such software could not be used easily with other non-traditional display shapes. Therefore, applications used with these circular displays would not be suitable for use with typical rectangular displays.

The DiamondSpin framework [28] offers support for circular tabletop multi-touch displays. Instances of a circular display used by the framework are achieved through either projecting the system's visual output onto a circular surface or occluding sections of the output. DiamondSpin supports these displays through several applications which are designed specifically for use on a circular tabletop interface; the framework provides several features to support applications intended for circular interfaces, such as the ability to orientate items towards the nearest edge of the display. Some of the example applications for this framework can be used with both rectangular and circular interfaces. Using the framework's features, the applications are able to appropriately arrange the layout of content to the display shape used. However, this adaptive ability is limited to only two regular display shapes: rectangular and circular.

The circular dashboard liquid crystal display [3] is a typical example of a technology designed with the intention of supporting ubiquitous computing [35]. In an era of ubiquitous and pervasive computing, more systems are being designed to fit naturally into their surroundings [9]. Thus, for computer systems to be ubiquitous they are required to be designed around typical user environments, not vice versa. The implication of this is that many previous standard elements of typical computing systems need to be reconsidered, such as the shape of a system's interface. One of the strengths of tabletop systems are their ubiquitous nature [29]; the ability for a tabletop system to manage different display shapes enhances this strength, allowing interfaces to better fit their environment.

Previous work that discusses the possible effects of displays with non-rectangular display shapes [33] highlights the benefits it may offer; these benefits include potential improvements to collaboration around the display between users. Vernier et al. [33] proposed a circular tabletop interface which would allow each user to have an equal share of the display. Though focused on a circular tabletop displays, they highlight the effect that a different display shape could have on the use of an interface; benefits noted in the work include the improved management of users' personal space.

It is thus evident that the visual content of systems will need to accommodate the use of different display shapes. A natural way to achieve this is to define specific layouts in software's visual content for each potential display shape it may be used with. This is the approach used with the software systems discussed thus far [10, 28]; however, for software which may have a wide range of potential interface shapes this would require significant extra development work.

The structured literature review [15] that informed this research project highlighted that there has been little innovation in utilising different display shapes in the past. This is likely due to the cyclical nature of dependency between display technology and its supporting software. However, recently the research activity in this area has started to increase, with research appearing that details investigations on adapting text to non-rectangular displays [26] and how visual content is best presented on legacy interfaces [27]. The outcomes from these pieces of research are a set of guidelines which should be followed to make sure content can fit different display shapes. These guidelines can be restrictive and place additional responsibilities on software developers and content designers. This, much like designing for unique content configurations for specific display shapes, will result in potentially significant additional development and design work.

A potential method to reduce this would be to use a tool which can adapt the layout of visual contents to different platforms, such as GUMMY [18]. This tool can take an existing layout as defined by a developer and quickly adapt it to the parameters of a specific platform, including the restrictions of the display. For example, the spacing of a predefined layout may be reduced by the tool when the target platform is known to use a small display. By implementing a method which allows the tool to dynamically adapt a layout of content items to different display shapes, software developers could produce multiple layouts for a piece of software to use with different display shapes relatively easily.

As opposed to using predefined layouts for each potential display shape, software could be designed to use a single layout which is adapted to fit different display shapes automatically. For example, SUPPLE [8] is a system which can automatically adapt the layout of contents to fit the parameters of a display, with the content automatically arranged by the system to fit various display sizes and resolutions. Furthermore, SUPPLE has the ability to adapt its visual contents based on input type. However, the system assumes that despite the changes in an interface's size, resolution, input type and colour support, it will always be rectangular. Changes to layout adapting systems, like SUPPLE, will be needed to allow them to adapt their layouts to non-rectangular displays.

There exists initial research into software capable of adapting its visual contents to different display shapes. Waldner et al. [34] discuss work carried out concerning the development of a system which allows windows to be adapted to make use of unusual display shapes. The unusual display shapes discussed by Waldner et al. consist of a series of overlapping projections which form the system's output, with these overlapping projections not always forming a rectangular shape. Therefore, it is important for the software used not to be dependent on a rectangular visual output. The technique presented works by automatically identifying the best location for new windows. An importance value is used for each pixel of the output and the magnitude of this value is used to assess its suitability to displaying a new window. The higher the value is, the less suitable the pixel is; pixels outside the output's shape are given a maximal importance value. This means that no window can be placed anywhere which will result in it occupying these pixels. This prevents content windows being placed where they may be partially occluded due to some of their regions existing outside the display area, ensuring that no visual content is obscured.

Waldner et al.'s technique appears to be beneficial for use with windows which have no layout defined between each other. The technique could be employed in other systems to manage the placement of visual content items when used with different display shapes. However, for visual content which may have a significant locational relation this technique could be considered unsuitable; for example, the order in which windows are created will influence where the technique positions them. Therefore, the technique currently has no method to accommodate for any locational relationship which are intended to exist between windows. If two visual content items are intended to be in close proximity to each other, there is no guarantee that the technique will position them together. As the positioning of content items can imply their functional relations to the user [4], the loss of these intended locational relations could be undesirable for some systems.

The potential benefits [9, 33] and growing supporting technologies [3, 7] of non-rectangular displays indicate that there is an increasing demand for systems to support them; more specifically, there is a growing demand for these different display shapes in tabletop systems [10, 28]. For tools which are used to design the layout of software [18] or systems which dynamically update the layout of visual content items [8], a method of adapting visual content layouts to different tabletop display shapes will be needed.

## 3 Identifying Display Shape Dependence Issues

For developing a method of adapting visual content layouts to different display shapes, the effects of applying a new display shape to a system's visual output need to be considered. Issues can occur when a display shape which differs from the shape of a display assumed by the software is used. In this section we define six of these *Display Shape Dependence Issues (DSDIs)*.

The DSDIs presented in this work are entirely novel; they were derived from first-hand observations on attempts by the authors to manually fit visual contents to different display shapes and from observations discussed in related work of Serrano et al. [26, 27]. Figure 1 groups the DSDIs into two groups: primary and secondary. *Primary DSDIs* are issues which can occur when displaying content intended for a particular display shape on a display of another shape with no modification. *Secondary DSDIs* are issues which can occur when performing modifications when displaying content intended for a particular display shape on a display of another shape, such as when attempting to resolve any primary DSDI. Note that the numbering or ordering of the DSDIs holds no significance and is only use for identification purposes.

*i)* ***Cropping DSDI:*** *Content items appear cropped as regions of them become positioned outside the display area.*

Cropping can have an effect of making the visual content items unfit for purpose; this is because the

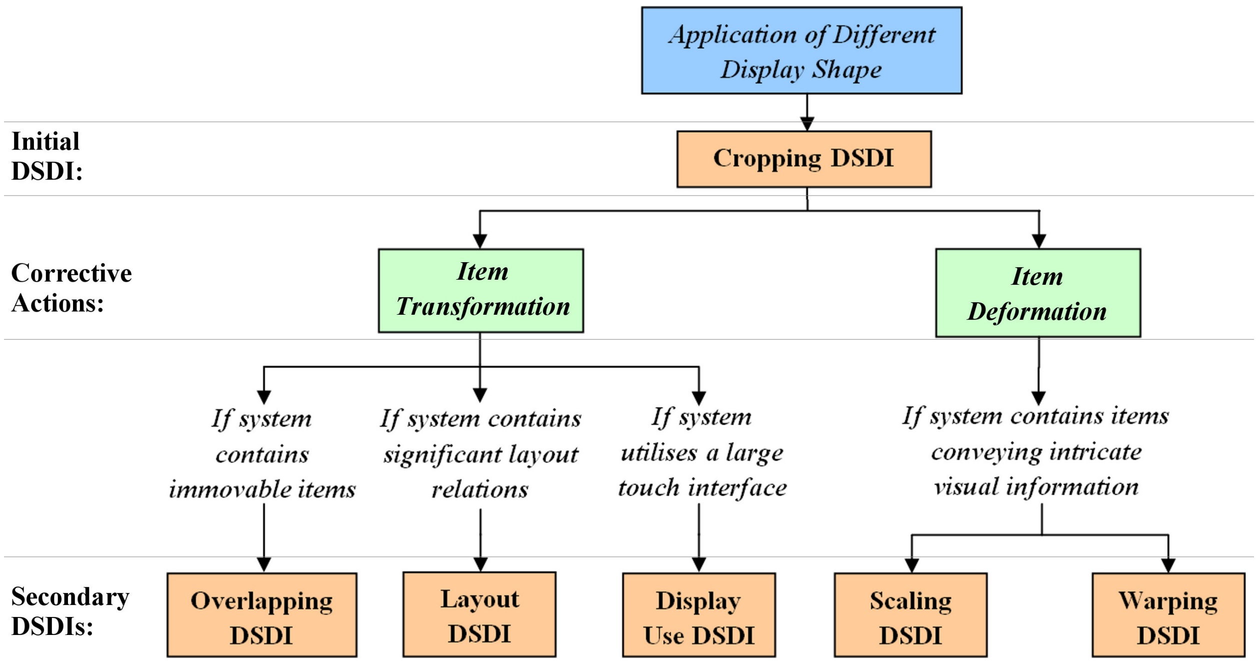
Application of Different Display Shape
Initial DSDI:
Cropping DSDI
Corrective Actions:
Item Transformation
Item Deformation
If system contains immovable items
If system contains significant layout relations
If system utilises a large touch interface
If system contains items conveying intricate visual information
Secondary DSDIs:
Overlapping DSDI
Layout DSDI
Display Use DSDI
Scaling DSDI
Warping DSDI

**Fig. 1** A flow chat mapping the causality of Display Shape Dependence Issues (DSDIs).

visual information they convey is lost. Users may also be unable to interact directly with cropped items. The cropping of visual content items outside the display area was also noted by Waldner et al. [34]. Therefore, methods of resolving this cropping would be required to make software capable of adapting to different display shapes. Attempts to resolve this cropping can result in secondary DSDIs.

*ii)* ***Overlapping DSDI:*** *Occlusion from other visual content items.*

Attempts to resolve the *Cropping DSDI* can result in further issues such as the movement of content items, which did not previously overlap, to locations where they cover each other. This overlapping of content items can result in regions of the visual content item becoming obscured. This has the same negative impact as the *Cropping DSDI*. Overlapping content items is not always an issue since the obscuring items could be moved by the user, but in systems where these items are immovable the occlusion would be irreparable.

*iii)* ***Layout DSDI:*** *The loss of layout relations between visual content items.*

The changing of content item positions could be an issue in applications where layout is important; this is because the positioning of content items can imply their functional relations to the user [4, 16]. For example, controls in an application may be placed next to the items they influence. If this layout relation is lost then it may become users what items the set of controls will influence. The loss of layout may not make content unfit for purpose but could be highly undesirable for software where the layout is intended to aid the user. However, loss of layout is not always considered an issue meaning this DSDI will only apply to specific content, tasks or applications.

*iv)* ***Scaling DSDI:*** *The scaling of visual content items to an extreme.*

Scaling – when a content item is resized while maintain its aspect ratio – a visual element may be performed to avoid the cropping of content. If this scaling is performed to an excessive magnitude, the visual information displayed by a content item may become incommunicable. For example, if a content item displaying text is scaled down to an extreme value the characters it displays may become too small and be illegible to users.

*v)* ***Warping DSDI:*** *The warping of visual content items.*

Warping is where the contents of a system are stretched and/or squashed in a non-uniform manner [19]. This could be applied to the entire visual output of a system to make it fit a particular display shape. However, this results in content items being stretched and squashed non-uniformly. Even when the magnitude of this deformation is relatively small it can make any visual information a content item displays incommunicable to the user. However, this DSDI may not apply in all circumstances; some visual content items may display visual information which is communicable to the user no matter how severely it is deformed. For example, if a content item's visual information is its colour – if the item can be seen then it is successfully communicating its visual information to the user.

*vi)* ***Display Use DSDI:*** *Display regions unused by the layout of visual content items.*

As a result of attempting to resolve the initial cropping, visual content items can often be translated to the same regions of the display. This can leave large areas of an interface unused. While this may not make the software unfit for purpose it is not desirable for large tabletop interfaces. This is due to the fact that on larger interfaces some regions of the display may be beyond a user's reach. If the display uses direct touch interaction, then items in these regions become isolated from the user's influence. There are solutions to this problem, such as users changing their positions around the interface or controls which can be used to indirectly manipulate out of reach items [23]. However, if all the content items are constrained to a single region of the display beyond a user's reach, the need to repeatedly use these solutions is undesirable. Changing positions to access remote content may not be possible in some environments due to the placement of the interface, for example, multi-touch interfaces may have several users positioned around them. Users interacting simultaneously with a single interface often have a tendency to establish territories [25]. A user wishing to move with these types of interfaces may disrupt others interacting with the system. Solutions involving the remote control of out of reach content items result in users manipulating content items through a proxy [30]. This means that their interaction would no longer be direct, which can be undesirable because it counteracts the benefits of direct touch interaction [24]. Ensuring content items are capable of being spread out across the interface when possible reduces the likelihood that all the items will be placed beyond the reach of the users.

For a tabletop software system to be capable of dynamically adapting its visual content to different display shapes, the *Cropping DSDI* needs to be resolved in such a manner that will minimise the influence of any of the potential secondary DSDIs. Figure 1 maps the causality of the DSDIs and notes the circumstances in which they may have undesirable consequences on a system's visual content. In the next section, using this list of DSDIs, we present a technique to adapt software to different display shapes in such a way that reduces the possibility of software becoming unusable due to these issues has been developed.

## 4 Minimising Display Shape Dependence Issues

A novel technique was developed that would resolve the *Cropping DSDI* and would minimise the effect of any secondary DSDIs. This technique is a method of adapting visual content items dynamically to different display shapes and comprises of two stages. The first stage of the technique is built on the concept of taking the original visual output of the software and transforming it to fit within the borders of a new display shape. This stage is referred to as *Virtual Projection*. The second stage of the technique uses the difference between the shape of virtual projection and the display shape to ensure that the layout can, if needed, utilise all regions of the display. This stage is referred to as *Position Pulling*.

### 4.1 Virtual Projection

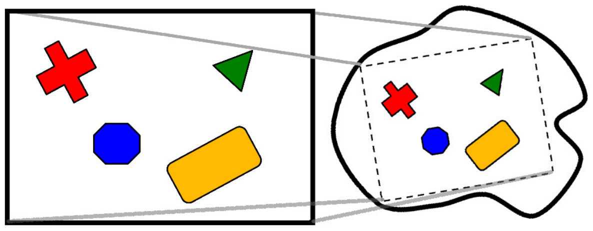

**Fig. 2** An example of the *Virtual Projection* stage.

For the first stage of the technique, the original shape of the software environment's visual output is treated as a single item. This item, referred to as the virtual projection, is translated, rotated and scaled so that it fits within the display shape being used. The transformations applied to the virtual projection are applied to any visual content items displayed in the software's original layout.

This stage of the technique has the effect of keeping the visual content items aligned with the virtual projection. This ensures that visual contents will not incur occlusion from lying outside the borders of the new display shape. This is because the content items will all be contained within the virtual environment which fits inside the display. Figure 2 demonstrates how the virtual projection and its contents are transformed together to fit a new display shape.

As the original visual output of most current software is likely to be rectangular, the virtual projection for most systems can assumed to be a rectangle. Finding the largest rectangle which can be placed within the display is required for this technique to utilise as much of the display as possible. The concept of finding the largest rectangle inside a shape is a known as the '*largest empty rectangle problem*'or the '*maximal rectangle problem*'; there are several existing solutions to finding the maximal rectangle which can be utilised [1, 20].

Methods also exist for finding the largest of any specific polygon in the display shape [31]. This allows for the adaptation of any software environment which may not have originally been rectangular. Therefore the technique can be used to adapt software originally designed for any specific display shape to other display shapes. As all the content items are translated, rotated and scaled by the same values their layout relations are preserved preventing the *Overlapping DSDI*. As no warping deformations are used in this technique, there can be no occurrence of the *Warping DSDI*. It is important to note that because of the scope of this work is limited to tabletop displays, the rotation of the content is not considered an issue. By definition, tabletop interfaces are horizontal allowing them to be accessed from any direction making the impact of orientation negligible.

There is the potential for issues with the *Scaling DSDI*; if the virtual projection is significantly smaller than the initial layout's environment, then the contained content items will be scaled to an extreme value. Indeed, it is possible that the content items may be scaled to an extent that they become unfit for purpose. To counter this, the virtual projection technique adheres to scaling limits. For systems where content items cannot be scaled to extremes, its content items should only be scaled to a predetermined limit. This may result in sections of the content items becoming occluded from overlapping each other or from the display edge because they cannot be made small enough. Without scale limits, the *Cropping* and *Overlapping DSDIs* are guaranteed not to occur but the *Scaling DSDI* remains a risk. With scale limits the potential risk of DSDIs are reversed. When utilising this technique, developers must decide on DSDI trade-offs in their software.

This stage of the technique is similar to previously presented methods of dynamically adapting content to different displays which entails constricting the placement of content to a specific region of the displays used [5, 22]. This region is recreated for every display shape using the same dimensions, ensuring the contained content is kept relative to the region in layout, scale and orientation.

This stage of the technique does have the potential to minimise the effects of most of the DSDIs. However, this stage of the technique does not counter the occurrence of the *Display Use DSDI* since only a single region of the display, the virtual projection, is used.

### 4.2 Position Pulling

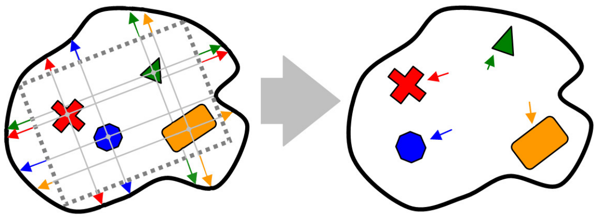

**Fig. 3** An example of the *Position Pulling* stage.

For the second stage of the technique, the gaps between the virtual projection and the display edges are utilised. The objective of this stage is to move content items into areas of the display which are left unused by the virtual projection. The edges of the virtual projection exert a 'pulling force' on content items. The magnitude of this force is determined by the size of gaps between the display border and the virtual projection edges. The magnitudes of these 'pulling forces' are also influenced by a content item's proximity to the edge instigating the force. Content items are pulled from their centroids to keep calculations simple; this means the shapes and sizes of the items moved are ignored.

The position pulling stage results in pulling content items into unused regions of the display (see Figure 3). Items positioned near large gaps between the virtual projection and the display borders are pulled into these areas. The left hand side of Figure 3 shows the 'pulling forces' which result from the gaps between the virtual projection area edges adjacent to the content items. Figure 3's right hand side shows the translation of the content items resulting from the combined influence of the 'pulling forces'.

## Algorithms

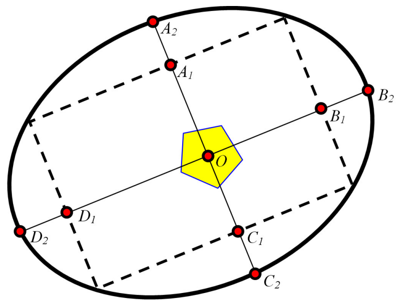

**Fig. 4** Detailed view of the information used in *Position Pulling*.

**Algorithm 1** Calculating the 'pulling force' vectors.

```
getPullVectors($O$, $A_1$, $A_2$, $B_1$, $B_2$, $C_1$, $C_2$, $D_1$, $D_2$)
    $\overrightarrow{LEFT} \leftarrow (\overrightarrow{D_1D_2} * (|\overrightarrow{OD_1}| \ / \ |\overrightarrow{B_1D_1}|)$
    $\overrightarrow{RIGHT} \leftarrow (\overrightarrow{B_1B_2} * (|\overrightarrow{OB_1}| \ / \ |\overrightarrow{B_1D_1}|)$
    $\overrightarrow{UP} \leftarrow (\overrightarrow{A_1A_2} * (|\overrightarrow{OC_1}| \ / \ |\overrightarrow{A_1C_1}|)$
    $\overrightarrow{DOWN} \leftarrow (\overrightarrow{C_1C_2} * (|\overrightarrow{OA_1}| \ / \ |\overrightarrow{A_1C_1}|)$
return $\overrightarrow{LEFT}$, $\overrightarrow{RIGHT}$, $\overrightarrow{UP}$, $\overrightarrow{DOWN}$
```

**Algorithm 2** Applying the 'pulling force' vectors.

```
applyPull($item$, $\theta$, $\overrightarrow{LEFT}$, $\overrightarrow{RIGHT}$, $\overrightarrow{UP}$, $\overrightarrow{DOWN}$)
    $\overrightarrow{PULL} \leftarrow \overrightarrow{LEFT} + \overrightarrow{RIGHT} + \overrightarrow{UP} + \overrightarrow{DOWN}$
    $\overrightarrow{PULL}.rotateBy(\theta)$
    $item.translateBy(\overrightarrow{PULL})$
return
```

Algorithm 1 shows how each vector representing the individual 'pulling force' is calculated and made proportional to the item's proximity to the corresponding edge. Algorithm 2 details how pulling force vectors are attained, where $\theta$ represents the orientation of the virtual projection. The existing values used in Algorithm 1 and Algorithm 2 correspond to those shown in Figure 4.

This stage of the technique effectively deforms the layout of the content items by warping it to fit the display shape. Though the layout is warped, the content items are not deformed in any way by this stage of the technique. The deformation of the layout ensures that more of the display's real estate is utilised and that the layout is not constrained to a small region of the display. This minimises the potential effects of the *Display Use DSDI*. The shape of the item does not influence the pull vectors, only the point it uses for translation. Taking the shape into account would require significantly more processing and would only result in a marginally more accurate pull vector.

The 'pulling forces' used in this stage of the technique assume the original environment of a piece of software is rectangular. Assuming a rectangular environment can be considered acceptable due to the current prevalence of software intended for use with rectangular displays [32]. However, it is possible that in the future the technique may need to adapt software designed for non-rectangular displays to different shapes. The virtual projection stage can easily adapt to this through using the maximal polygon opposed to the maximal rectangle. For the position pulling stage to utilise content intended for a non-rectangular display, a vector representing the 'pulling force' for every unique edge of the virtual projection would need to be created. Vector addition between each of these vectors representing the 'pulling forces' then creates a single vector which can be used to translate a content item.

After employing the two stages discussed, the resulting technique has the potential to minimise the DSDIs identified, when different display shapes are utilised by a system. While the technique does has the ability to counter any DSDI, it does have the drawback of requiring trade-offs in its implementation of which DSDIs it minimises the consequences.

## 5 Implementation

The technique discussed in the previous section was implemented into version 2.1 of the *SynergyNet* framework [2, 17], a multi-touch software framework intended for use with rectangular tabletop interfaces. The framework functions by rendering a range of content items, such as images, text and video, which can then be manipulated by users. *SynergyNet* is built in Java, allowing it to be run across different operating systems and can accept a wide range of different multi-touch protocols such as TUIO [14]. This makes the framework capable of being used in a wide range of systems with different displays. The framework's ability to adapt to different systems makes it a suitable candidate for use with different display shapes. Another reason for this choice in framework was due to its easily accessed library of supported open-source applications. Since the applications are produced by a number of different developers for different purposes, there is a wide range of content items and layouts available. Being implemented at the framework level allows any of the supported applications to potentially utilise different display shapes.

To utilise the technique the software needs to be informed of the dimensions of the display shape currently available. This is done through the creation of virtual counterparts to the display shapes utilised by the system. These virtual counterparts are represented as vector files which are loaded into applications through the framework's configuration interface. The approach of using virtual counterparts allows each display shape representation to store information on the positioning of their maximal rectangle, or polygon. The maximal rectangle can be used by the virtual projection stage of the technique because all the *SynergyNet* applications are currently built using a rectangular environment.

The technique is implemented in such a way that it is always employed when an application attempts to transform a content item using absolute values. This means that, in addition to the initial positioning of content, the technique is employed when an application needs to transform an item to a specific position. For example, if a new item is to be added to the 'centre' of the visual content the technique would be applied to ensure it appears in the middle of the maximal rectangle, even if the item is added sometime after the application's initialisation. The technique does not need to be utilised for relative transformations, such as those used when adjusting an item's position, scale or rotation in response to a user gesture.

The *SynergyNet* framework was adapted to apply a range of borders which would allow the display to emulate different shapes. These shapes were chosen after consultation with a furniture manufacturer who had experience with designing multi-touch tables. The manufacturer was in a strong industry position for making predictions concerning the likely design features of future multi-touch tabletops, including their display shapes; the shapes chosen are shown in Figure 5.

Some of these shapes were derived from common table shapes which could be used with tabletop interfaces, such as the semi-circle design. This allows for users positioned around the display in various places to have an equal share of the interface which can be beneficial to collaboration. The circle shape was also chosen for its ability to allow users to have equal shares of the display. The intersecting circles shape was designed to make further use of multi-touch's facilitation of collaboration by providing focal points and areas to share content items in. The larger circle of the display shape can be used for one activity such as displaying content items whereas the smaller circle could be used for another activity such as manipulating the items. The triangle was chosen because of its regular nature, meaning that all its sides and corners are congruent.

## 6 Observations

With the technique implemented into the *SynergyNet* framework four applications were used in conjunction with the range of different display shapes to observe the impact of the technique. These applications were created by the primary author and other developers for previous studies in the *SynergyNet* project [11] The applications were chosen because of their differing methods of organising their visual content items and the varying significance of the layout of their items. This was because the authors wish to investigate how the technique will impact the arrangement of both items with and without a significant layout. The four applications used were:

- **XML Puzzle**: Reads XML files to generate questions and answers that are randomly placed on screen which the user then has to pair up.
  - All the content in this application is positioned randomly.
  - There is no significance to the layout of any of the contents.
- **Collage**: Users move visual content items about on top of a large background image to make a scrapbook-type collage of which they can then save a snapshot.
  - All the content in this application is positioned randomly except for the background which is specifically located in the centre.
  - There is no significance to the layout of the contents other than the background which must be central.
- **Grid**: Nine items are positioned in a grid layout which users can reposition, rotate and scale through multi-touch gestures. This is one of the applications supported by the *SynergyNet* framework which help familiarise users to interacting with a large multi-touch interface.
  - All the content in this application is specifically positioned.
  - The locational relation between all the items in this application is strong.
- **Simple Map**: Users supply countries in a configuration file and the application will show a world map when started with the specified countries' flags positioned on top of them; users can then move the flag items around the map.
  - All the content in this application is specifically positioned.

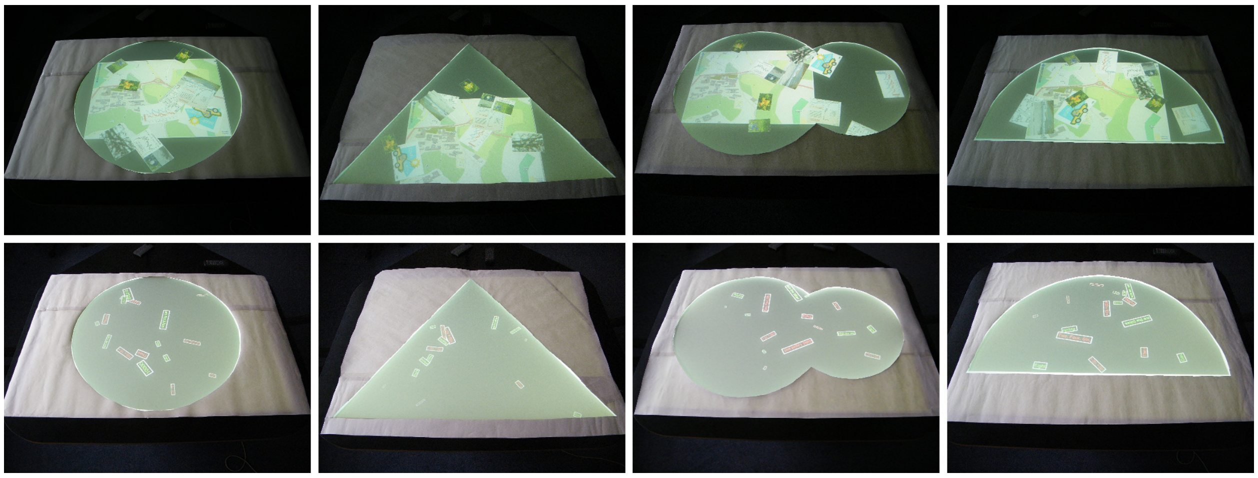
**Fig. 5** Several applications in use with different display shapes.

– The locational relation between the flag items and the background image is strong.

The different display shapes were emulated by covering parts of the screen used. Figure 6 shows all of the applications chosen to be used with the technique for observation. In the next section, observations from the use of these applications were made on how the adaptation of their content to fit the new display shapes affected their initial usability. The study focused on applications in which the layout of content was significant in some way.

After the application of the technique, the *XML Puzzle* and *Collage* applications were able to be utilised as normal with the DSDIs totally resolved without any observable issues or additional complexities arising. This is likely because there is little-to-no significance to the layout of both applications' visual content items. The background image in the *Collage* application was the only item with significant positioning in either. This background image effectively represented the maximal rectangle as only the virtual projection stage of the technique was applied to it.

However, the two applications reliant on the layout of their visual contents highlighted a number of complexities to applying the technique; these were the *Grid* and *Simple Map* applications. The complexities arising from these two applications are important to consider due to the fact that the *Grid* and *Simple Map* applications bear the strongest similarities to the majority of legacy software. The entirety of the contents of these two applications are specifically positioned, much like how the contents of traditional GUIs in legacy software are.

As can be seen in Figure 7, all the items are visible and there is no cropping in the initial layout. However, when any of the four example display shapes are applied, several content items are occluded to varying extents. When the virtual projection stage of the technique is applied the cropping is resolved. At this stage of the technique, the *Display Use DSDI* can be observed due to the presence of unused regions of the display.

The position pulling stage of the technique can be seen in Figure 7 to pull the content out into these previously unused regions of the display. However, the layout of the content items is deformed as a result of this stage. While acceptable for the *Grid* application, this is problematic for software which cannot tolerate the influence of the *Layout DSDI*.

It was observed from the impact of the technique that developers may wish for some content items to be affected by it in different ways. For example, a content item may be intended to act as a background to an application. If the implemented DSDI-minimising technique a system uses to adapt to different display shapes is unaware of this, the item would be scaled with the virtual projection. This would have the effect of the background being constrained to a single region of the display, leaving other areas of the interface blank without a background. In the implemented system, a method of making content item exempt from the influence of specific stages of the technique was made available.

In Figure 8, the map was made exempt from the influence of the position pulling stage of the technique. If the map item was made exempt from both stages of the technique, the map would remain in the same position for all display shapes. Though this would

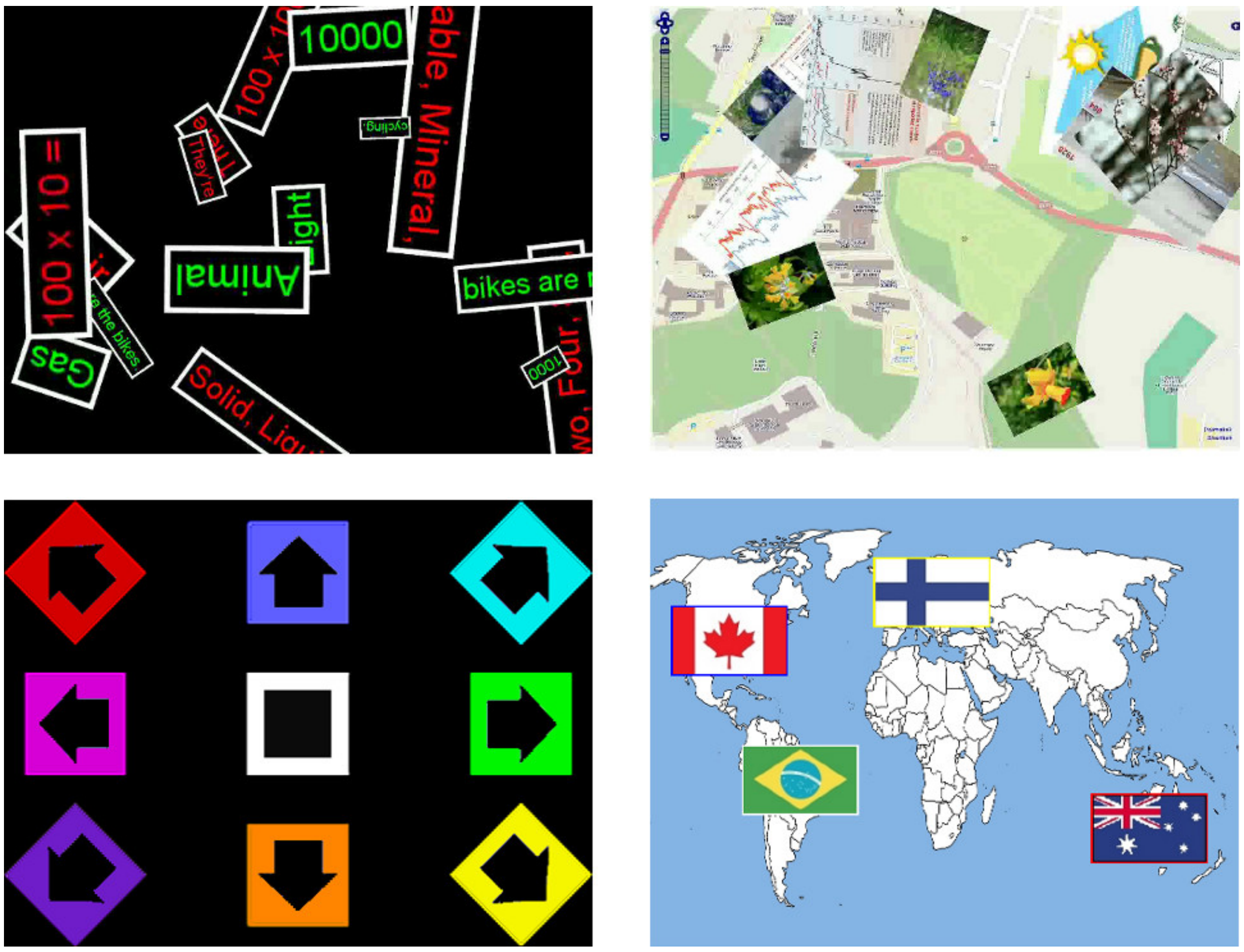

**Fig. 6** The applications chosen for observation, from left to right: *XML Puzzle*, *Collage*, *Grid* and *Simple Map*.

guarantee that the map would remain a background to the entire display, areas of it would be occluded which may not be desirable. Therefore, the map item was made to adhere to the virtual projection stage of the technique, but not the position pulling stage. As a result of this, the map effectively represents the virtual projection on each display shape. The placement of the virtual projection on the circle and intersecting circles display highlights how the contents may be rotated to fit the display shape. Since the software is intended for use on a tabletop display this is not an issue, because the horizontal nature of the display signifies that there is no single vantage point that the interface will be viewed from.

Figure 8 demonstrates the importance of the layout of content items. Up to the virtual projection stage of the technique, the flag items correctly align with the countries on the map; however, the position pulling stage deforms the layout of the flags in such a way that disengages this alignment. If it is important for this alignment to be kept then it is best if the flags are made exempt from the position pulling stage of the technique. As a consequence of this, however, areas of the display will be left unused by the layout. It therefore becomes a choice developers must make between whether to preserve the layout of content items or to maximise the usage of the display's real estate. A clear example of this choice can be seen in the technique's influence on the *Intersecting Circles* display shape in Figure 8 – without the influence of the position pulling stage of the technique, a large region of the display is left unused by the initial layout, resulting in the *Display Use DSDI* occurring. However, with the influence of the position pulling stage of the technique, several of the flag items appear off the map, away from their respective positions, indicating the *Layout DSDI* has occurred.

From the use of the implemented technique it was noted that in certain scenarios the limits imposed on the scaling of content items by the technique could

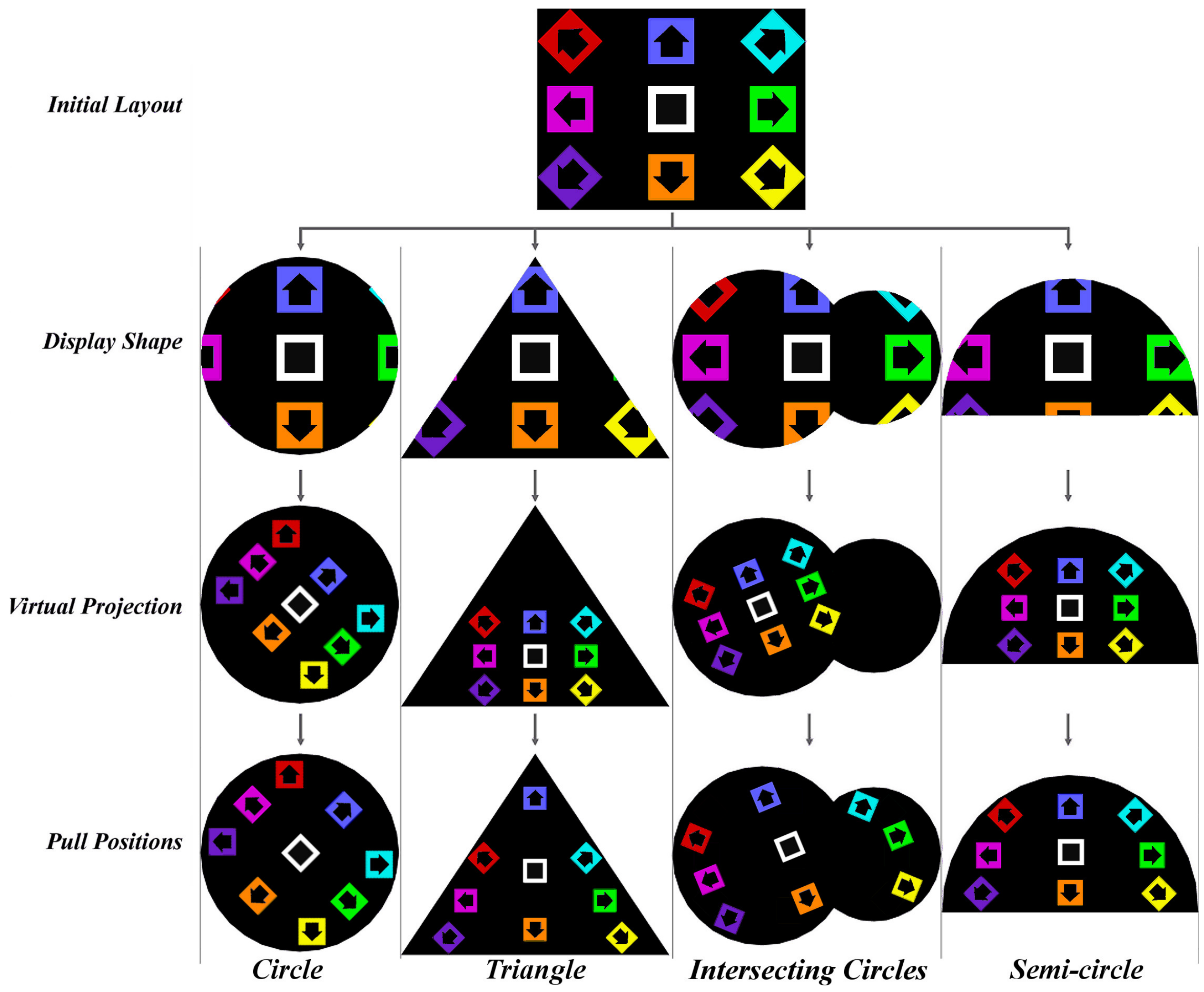

**Fig. 7** Influence of the implemented technique on *SynergyNet*'s *Grid* application.

cause the *Overlapping DSDI* to occur. If content items positioned close to each other are scaled to their predetermined limits but the virtual projection, and therefore their layout, is scaled beyond this then occlusion may occur. This occlusion is caused by the content items overlapping. A potential adaptation of the virtual projection stage of the technique could resolve this. This adaptation makes use of the smallest box possible which encloses the content items, rather than using the shape of the entire original environment. Any unused regions of the original display between the content items and the display's edges are ignored. This variation of the technique could be used with any software where the distance between content items and the display edges does not need to be preserved. By discarding the unused regions of the original display surrounding the content items, the virtual projection is likely to be smaller. Therefore, the scaling required to fit the virtual projection into a new display shape will be reduced. This diminishes the likelihood of content items and their layout being scaled to beyond their limits.

Figure 9 shows the options available to developers utilising the technique. Choices made in its implementation, such as which stages of the technique are employed and the constraints placed on these stages, can influence which DSDIs it prevents and counters. For example, if a developer wishes a content item to act as the background to the entire display, and its cropping is not an issue, then both stages of the technique can be circumvented. For content items not set aside to be background images, developers can choose whether or not to apply the position pulling stage of the technique. If applied, the content items will maximise the use of the display shape's real estate by being pulled into previously unused areas of the

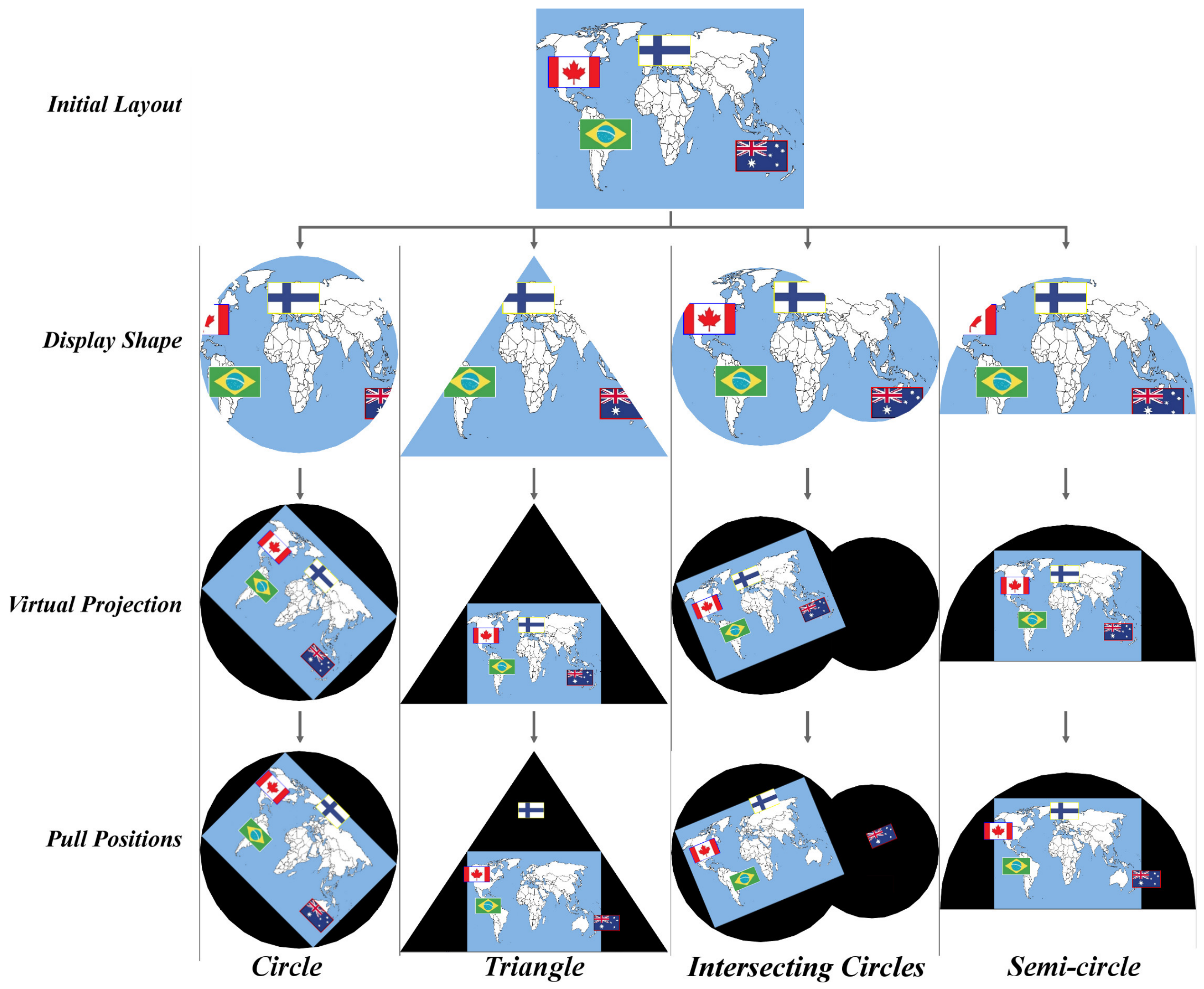

**Fig. 8** Influence of the implemented technique on *SynergyNet*'s *Simple Map* application.

display. However, if the layout of the content items is of a greater importance than the maximisation of display usage, the developer can choose to disregard the position pulling stage for any content items in the layout.

In the *SynergyNet* implementation, both stages of the technique are, by default, applied to content items whenever they are transformed using absolute values. However, application developers using the framework have the option to override this for specific content items. For example, they can apply changes to the technique, such as applying scale limits to the virtual projection, to all content in an application or to individual items. The ability to choose which stages of the technique affect a content item gives developers control over its influence. Without this ability there is still a danger of unacceptable DSDIs occurring in software despite the presence of the implemented technique. This highlights the importance of any technique intended to minimise the influence of the potential DSDIs in a system not only to be capable of adapting to different display shapes, but to be also capable of adapting to the needs of the developer. This allows developers to make informed, user-centric design and implementation decisions on how content should be displayed.

## 7 Conclusions and Future Work

The development of the techniques presented in Section 4 has shown that when designing software for use on differently shaped tabletop displays, the *Cropping DSDI* must be considered. In addition to this, secondary DSDIs which can result from attempts to resolve the initial cropping must also be addressed. Developers should also consider which DSDIs their software can tolerate and which must be handled by

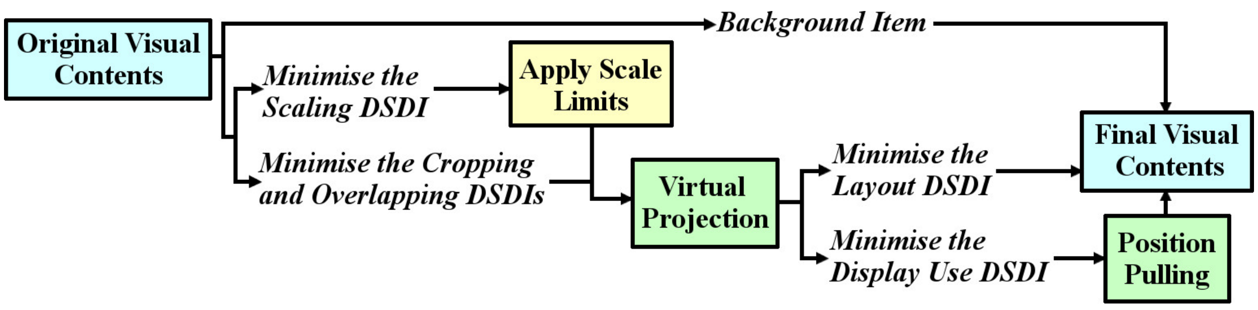

**Fig. 9** Flowchart demonstrating navigation of the technique.

a DSDI-minimising technique. Choices must be made between whether to preserve the layout of content items or to maximise the display usage.

Developers must consider the implementation of any proposed method of countering the DSDIs. The identification, or creation of, an adequate DSDI-minimising technique which allows a particular system to adapt to different display shapes could form part of design guidelines for emerging technologies. It is important to note that not only the list of DSDIs for a system should influence the design of a DSDI-minimising technique; the need for applications to control the magnitude of a DSDI-minimising technique's influence is also important to consider.

Furthermore, it is important to note that although the technique presented in this work is a solution to the DSDIs of tabletop interfaces, it is not always guaranteed to be the best solution. It may be possible in many combinations of content and display shapes to find solutions where the impact of the steps taken to avoid DSDIs is less than the impact of the presented technique. The solution of less impact may be preferable for developers, as it will allow content to appear more consistent across different displays, allowing for the software to appear more intuitive and familiar for users across devices. Identifying what solution is best suited for a given situation should be at the centre of any future research involving this topic. Despite no guarantee that the technique is the optimal solution for a given situation it still is a significant step forward for resolving DSDIs as it can (i) always supply an adequate solution to a wide-range of scenarios; (ii) be automated as part of an implementation as shown in this work; and (iii) allow developers some control over its execution.

There are a number of reasons why it is important that software becomes display shape independent. There is now a growing push for software systems to become ubiquitous with the increased availability of natural user interfaces; for a system to become ubiquitous it must fit its environment [9]. Being able to change the display shape of a system would aid in achieving this.

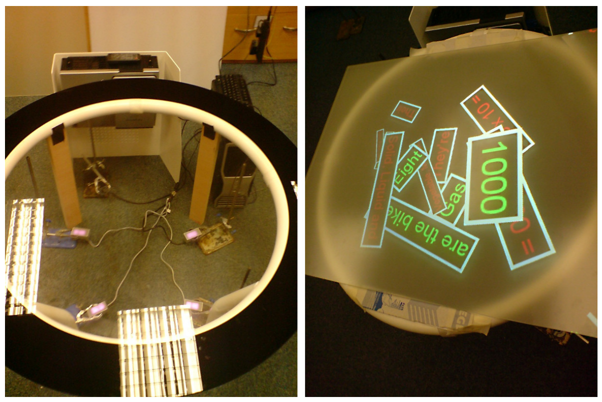

**Fig. 10** A prototype of a circular tabletop display.

The examples of non-rectangular displays discussed in Section 2 reinforces the need for their support. In addition to these, many devices brought to market in the recent growth of wearable devices utilise non-rectangular displays, specifically smart watches [13]. Figure 10 shows a prototype built by a leasing furniture manufacturer featuring a circular tabletop interface intended for use as an interactive kiosk for public spaces. The lack of readily available software for the interface's circular display meant that if the tabletop was to be used then bespoke software would need to be commissioned at a significant cost to the manufacturer rather than using any of the readily available off-the-shelf products intended for interactive kiosks. This highlights the need for making it a simple as possible for software to support different display shapes.

The technique presented in this paper can be adopted by developers to allow their content to fit different display shapes, much like the guidelines presented in the work of Serrano et al. [26, 27]. However, this technique deviates from the guidelines to provide developers with greater freedom and flexibility. Without encumbering software developers and content designers with additional restrictions the technique is able to resolve the DSDIs with minimal input. This allows for development time to be saved on adding support for different display shapes and also offers more freedom in the design of the content.

Furthermore, the technique outlined in Section 4 could be used beyond tabletop systems. However, it is possible for the DSDIs in other systems to vary and differ from those presented in this work [27]. For example, attempts to resolve the initial *Cropping DSDI* may require content items to be rotated. If a user is viewing the display from a single perspective, such as with a vertical display, then any change in content item's orientation is undesirable. Information displayed by visual content items, such as text, could appear upside down to the user, making it difficult to read. However, it is not an issue for systems which allow users to view from multiple perspectives around the display, such as those which utilise horizontal interfaces. Future work could entail the outlining of DSDI for other sets of systems and the production of techniques to minimise their influences.

Further research involving the technique could involve investigating how it functions with more extreme shapes. Although the shapes used in the implementation of the technique discussed in this study vary greatly and cover a wide range of what is likely to be used as tabletop display shapes, it is possible that shapes that vary from rectangular display in much more extreme ways could be desirable. Research utilising these extreme shapes could involve finding acceptable limits to place on parts of the technique such as maximum distances to pull content items.

While this work details the use of an implementation of the technique, the next step should be a full study investigating how the technique works for both real-world developers and end users. A study similar to that carried out by Serrano et al. [27] to review the impact of their proposed guidelines is likely to be a suitable approach. By instructing developers to utilise the technique when building some software, it should be possible to find out the suitability of the technique in real-world scenarios. We can then review the impact of the technique in these implementations to find if the results are suitable for end-users.

It is important to note that the scope of this work entailed resolving DSDIs relating to the layout of visual contents on tabletop interfaces. Future work has the potential of applying the technique presented to other kinds of interfaces. This future work would likely focus on additional restrictions the technique would have to take into account, such as the need for a specific orientation with vertical displays. Future work could also look at expanding the technique to resolve more than just DSDIs relating to the layout of visual content. Extra steps or adding limits to the transformations of elements within the technique may help improve usability and readability when applying content to a different display shape.

The techniques presented in this paper can be used in several scenarios. One such use of the technique is its implementation into the application layer of a piece of software; this would allow a specific layout of visual content items to be dynamically adapted to different display shapes. As this implementation of the technique would be customised to the application's specific layout, it would ensure the best results. However, this does have the drawback that the technique would need to be redefined for every unique application.

Another use for the technique is its implementation into an adaptable and extensible framework. In a framework-level implementation, the same technique can be employed by various applications to adapt their visual contents to different display shapes; this is similar to how systems like SUPPLE [8] adapt contents to display parameters. However, a drawback to this approach is that the technique's implementation may not be suitable for some applications. For example, if the technique implementation does not enforce scale limits, the *Scaling DSDI* may occur. This may be acceptable for some applications, such as those intended to use content which can still communicate information to the user at any size. However, it would be unacceptable for other applications, such as those using text. Therefore, it is important when implementing this technique at a framework level that applications can change how the technique influences them. The technique's use of higher-level functions, such as the ray firing used in the position pulling stage, make it unsuitable for implementation in lower level software.

The technique could also be implemented as part of a design tool which is used to adapt the layout of visual contents, similar to GUMMY [18]. Using this implementation, software developers can use the resulting tool to adapt a layout of content items to a specific display shape. Developers can then make any further changes manually if the tool's automatically generated DSDI-minimising technique is not adequate. Once a developer is satisfied with the software layout they can then specify for its use in their software when the appropriate display shape is present. This approach has the advantage of allowing developers to check that the layout will be suitable for a specific shape. However, it does potentially limit the display shapes the system can use to those the developer has input into the tool.

Finally, the technique could be transformed into a set of guidelines for developers to follow when creating software interfaces without the use of a design tool. The guidelines could also incorporate the decisions to be made concerning the trade-offs in which DSDIs are minimised. These guidelines could then be formalised [21], allowing their application to software to be automated in places. This would aid with the technique's integration into frameworks and design tools. With the technique implemented in a tool used to design a wider range of software, its influence can be tested on a wider range of user interfaces with varying complexities in future studies, especially in educational contexts [12, 17].

In summary, the techniques presented in this paper are capable of minimising the effect of the identified DSDIs, but require trade-offs. Developers using the technique must decide which DSDIs could not be tolerated by their software and must adapt the technique according. Future tabletop software development would benefit greatly from the creation of guidelines to be followed when adapting software to become display shape independent. The technique produced in this paper, or another similar DSDI-minimising technique, should form part of these guidelines.

## Acknowledgements

This work was partially funded under the UK's EPSRC/ERSC Teaching and Learning Research Programme (TLRP) *SynergyNet* project (RES-139-25-0400). The authors would also like to thank Professor Liz Burd and Dr Andrew Hatch for their supervision of the primary author's master's degree from which this work originally stems. The authors would also like to thank the members of the Durham University Technology Enhanced Learning Special Interest Group for supporting the redrafting of this manuscript. The source code for the technique's implementation discussed in this manuscript is freely available here: `https://github.com/synergynet/synergynet2.1`.

## References

[1] A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 278–290. ACM Press, 1987.

[2] I. AlAgha, A. Hatch, L. Ma, and E. Burd. Towards a teacher-centric approach for multi-touch surfaces in classrooms. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 187–196. ACM Press, November 2010.

[3] J. Boyd. Circular LCD debuts, IEEE spectrum. `http://spectrum.ieee.org/computing/hardware/circular-lcd-debuts`, 2007. [online; accessed 2017-05-20].

[4] L. L. Constantine and L. A. D. Lockwood. Software for use: a practical guide to the models and methods of usage-centered design. *ACM SIGCHI Bulletin*, 32(1):111–114, 1999.

[5] D. Cotting and M. Gross. Interactive environment-aware display bubbles. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, page 245. ACM Press, Oct. 2006.

[6] P. Dietz, R. Raskar, S. Booth, J. van Baar, K. Wittenburg, and B. Knep. Multi-projectors and implicit interaction in persuasive public displays. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 209–217. ACM Press, May 2004.

[7] M. D. Finney, M. W. Oliver, P. M. Pierce, and T. Sutherland. Communication device, US Patent No. USD600228, September 2009.

[8] K. Gajos and D. S. Weld. SUPPLE: automatically generating user interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 93–100. ACM Press, 2004.

[9] A. Greenfield. *Everyware: The dawning age of ubiquitous computing*. Peachpit Press, 2006.

[10] T. E. Hansen, J. P. Hourcade, M. Virbel, S. Patali, and T. Serra. PyMT: a post-WIMP multi-touch user interface toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 17–24. ACM Press, 2009.

[11] S. Higgins, E. Mercier, L. Burd, and A. Joyce-Gibbons. Multi-touch tables and collaborative learning. *British Journal of Educational Technology*, 42(6):1041–1054, 2011.

[12] A. Joyce-Gibbons, J. McNaughton, E. Tan, N. Young, G. Beauchamp, and T. Crick. *SynergyNet* into schools: facilitating remote inter-group collaborative learning using multi-touch tables. In *12th International Conference on Computer Supported Collaborative Learning*, 2017.

[13] Y. Jung, S. Kim, and B. Choi. Consumer valuation of the wearables: The case of smartwatches. *Computers in Human Behavior*, 63:899–905, 2016.
[14] M. Kaltenbrunner and R. Bencina. reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pages 69–74. ACM Press, 2007.
[15] B. Kitchenham. Procedures for performing systematic reviews. Technical report, Keele University, 2004. TR/SE-0401.
[16] J. McNaughton, T. Crick, and A. Hatch. Determining Device Position through Minimal User Input. *Human-centric Computing and Information Sciences*, 7(1):37, 2017.
[17] J. McNaughton, T. Crick, A. Joyce-Gibbons, G. Beauchamp, N. Young, and E. Tan. Facilitating collaborative learning between two primary schools using large multi-touch devices. *Journal of Computers in Education*, pages 1–14, 2017.
[18] J. Meskens, J. Vermeulen, K. Luyten, and K. Coninx. Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 233–24. ACM Press, 2008.
[19] T. Milliron, R. J. Jensen, R. Barzel, and A. Finkelstein. A framework for geometric warps and deformations. *ACM Transactions on Graphics*, 21(1):20–51, 2002.
[20] A. Naamad. On the maximum empty rectangle problem. *Discrete Applied Mathematics*, 8(3):267–277, 1984.
[21] D. Ngo, L. Teo, and J. Byrne. Formalising guidelines for the design of screen layouts. *Displays*, 21(1):3–15, 2000.
[22] R. Raskar, R. Raskar, J. Baar, J. Baar, P. Beardsley, P. Beardsley, T. Willwacher, T. Willwacher, S. Rao, S. Rao, C. Forlines, and C. Forlines. iLamps: Geometrically aware and self-configuring projectors. *ACM Transactions on Graphics*, 22(3):809–818, 2003.
[23] K. Ryall, C. Forlines, A. Esenther, F. Vernier, K. Everitt, M. Wu, D. Wigdor, M. Morris, M. Hancock, and E. Tse. Informing the Design of Direct-Touch Tabletops. *IEEE Computer Graphics and Applications*, 26(5):36–46, 2006.
[24] J. Schöning, P. Brandl, F. Daiber, F. Echtler, O. Hilliges, J. Hook, M. Löchtefeld, N. Motamedi, L. Muller, P. Olivier, T. Roth, and U. von Zadow. Multi-Touch Surfaces: A Technical Guide. Technical report, Technical University of Munich, 2008. TUM-I0833.
[25] S. D. Scott, M. Sheelagh, T. Carpendale, and K. M. Inkpen. Territoriality in collaborative tabletop workspaces. In *Proceedings of the 2004 ACM Conference on Computer-Supported Cooperative Work*, pages 294–303. ACM, 2004.
[26] M. Serrano, A. Roudaut, and P. Irani. Investigating text legibility on non-rectangular displays. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 498–508, 2016.
[27] M. Serrano, A. Roudaut, and P. Irani. Visual composition of graphical elements on non-rectangular displays. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4405–4416. ACM, 2017.
[28] C. Shen, F. D. Vernier, C. Forlines, and M. Ringel. DiamondSpin: an extensible toolkit for around-the-table interaction. In *Proceedings of the 2004 CHI Conference on Human Factors in Computing Systems*, pages 167–174. ACM Press, 2004.
[29] S. P. Smith, E. Burd, and J. Rick. Developing, evaluating and deploying multi-touch systems. *International Journal of Human-Computer Studies*, 70(10):653–656, 2012.
[30] S. P. Smith, E. L. Burd, L. Ma, I. AlAgha, and A. Hatch. Relative and absolute mappings for rotating remote 3D objects on multi-touch tabletops. In *Proceedings of 25th BCS Conference on Human-Computer Interaction*, 2011.
[31] G. T. Toussaint. Computing largest empty circles with location constraints. *International Journal of Computer & Information Sciences*, 12(5):347–358, October 1983.
[32] A. van Dam. User interfaces: disappearing, dissolving, and evolving. *Communications of the ACM*, 44(3):50–52, 2001.
[33] F. Vernier, N. Lesh, and C. Shen. Visualization techniques for circular tabletop interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 257–265. ACM Press, May 2002.
[34] M. Waldner, R. Grasset, M. Steinberger, and D. Schmalstieg. Display-adaptive window management for irregular surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 222–231. ACM Press, 2011.
[35] M. Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.

**James McNaughton** is a researcher at Durham University (UK) whose research interests include HCI, natural user interfaces and augmented reality. His current work involves investigating the use of emerging interaction technologies in classroom environments.

**Tom Crick** is Professor of Computer Science & Public Policy at Swansea University (UK). His research interests are interdisciplinary, data-driven and computationally-intensive, including data science, intelligent systems, software sustainability, public service innovation and STEM education.

**Shamus Smith** is a Lecturer in Computer Science at the University of Newcastle (Australia). His current research interests include touch-based technologies, mobile technology for augmented reality and the reuse of gaming technology.