Name:Neeraj Kumar

Email Address:

neeraj.kumar.cse06@itbhu.ac.in

neerajpkumar@gmail.com

Freenode IRC Nick:

I do not have a freenode IRC account.

Location (City, Country and/or Time Zone):

Varanasi,India. (GMT+5:30)

Proposal Name: **Desktop dock(Idea 2.1.6)**

Brief explanation**:** A *Mac OS X*-style dock containment.

Expected results: A containment that provides a similar user experience to the Mac OS X dock: application launchers that are also task bar entries when the application is active and a separate area for widgets such as the trash, battery, etc.

Motivation for Proposal / Goal:

I feel the need for eye candies, appreciate KDE delivering just tha, that too for free , and believe in open source. I have been working on GUI for a while , and have developed a similar part for an open source project called SPLINE which is a state-of-art, inter-operable ,Open Source , E-learning platform.The screen-shots have been attached.

I am sure that my involvement with the KDE development and Plasma will help me a lot , both in increasing my knowledge and contributing more to Open Source.

Implementation Details:

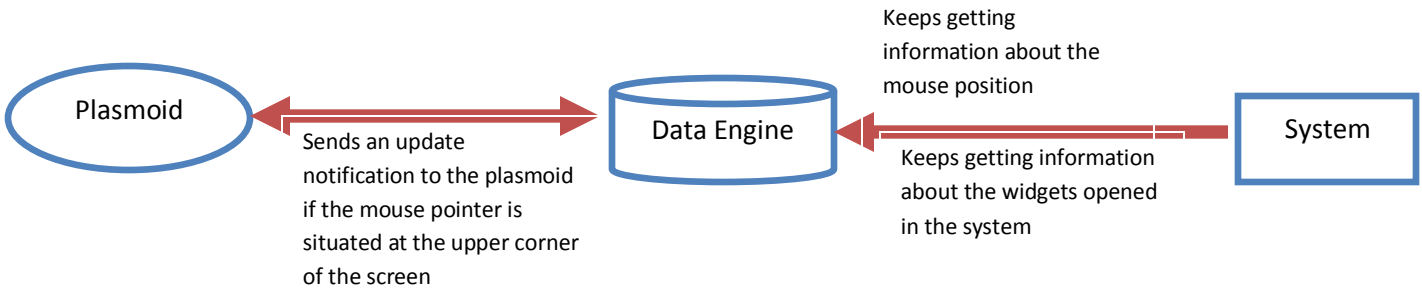
We know that Plasma uses a somewhat coined implementation of the Model View Controller framework .Where plasmoids are the views and the data engines server as the model for these views.

We create a plasmoid, which by default is hidden, and positioned at the top of the screen.We make a data engine for this plasmoid.The data engine has the following responsibilities:

1>Getting the mouse position every 333 milli seconds and sending an notification to the plasmoid.

2>If the pointer hovers on the upper part of the screen(say an area of (screen.width)x(100) pixels)then the plasmoid is notified to become visible.

3>The data engine also stores information about the currently running applications, and a reference to the widget of these applications.



A sample Code (More like an algorithm)

Dock.desktop

```
Name=Dock
Comment=MAC OS X Style Desktop
Type=Service
ServiceTypes=Plasma/Applet

X-KDE-Library=plasma_applet_dock
X-KDE-PluginInfo-Author=Neeraj Kumar
X-KDE-PluginInfo-Email=neeraj.kumar.cse06@itbhu.ac.in
X-KDE-PluginInfo-Name=plasma_applet_dock
X-KDE-PluginInfo-Version=0.1
X-KDE-PluginInfo-Website=http://neeraj.blogspot.com
X-KDE-PluginInfo-Category=Desktop Services
X-KDE-PluginInfo-Depends=
X-KDE-PluginInfo-License=GPL
X-KDE-PluginInfo-EnabledByDefault=true
```

Dock.h

```
// Include the custom header files

//Include all the plasma headers needed

class QSizeF;

// Define our Applet
class Dock : public Plasma::Applet
{
    Q_OBJECT
    public:
      // Basic Create/Destroy
      Dock(QObject *parent, const QVariantList &args);
      ~Dock();
      // Specific functions
      dataUpdated(DataObject *dOb);
      showDock(QPainter *p, const QRect &contentsRect);









        // The paintInterface procedure paints the applet to screen
        void paintInterface(QPainter *painter,
                const QStyleOptionGraphicsItem *option,
                const QRect& contentsRect);
       void init();

    private:
       //This will include all the objects which represent widgets ,which are
running in the system
       WidgetAlias widgetAlias[];
};
// This is the command that links your applet to the .desktop file
K_EXPORT_PLASMA_APPLET(KDEDock, Dock)

class WidgetAlias
{
       private:
              int processID;
              svg imageToShow;
       public:
              restoreWidget();
              //transfers the control of the screen to this widget
              closeWidget();
              //closes the application this object is an alias of
              minimiseWidget();
              //minimizes the application this object is an alias of
```

```
}


#endif
```

Dock.cpp

```
#include "plasma-tutorial1.h"
#include <QPainter>
#include <QFontMetrics>
#include <QSizeF>

#include <plasma/svg.h>
#include <plasma/theme.h>

KDEDock::Dock(QObject *parent, const QVariantList &args)
    : Plasma::Applet(parent, args),m_icon(null)
{
    setBackgroundHints(DefaultBackground);
    resize(400, 200);
}


Dock::~Dock()
{
    if (hasFailedToLaunch())
        {
        // Cleanup the initialized variables
        }
         else {
        // Save settings
        }
}

void Dock::init()
{

    // A small demonstration of the setFailedToLaunch function
    if (m_icon.isNull())
        {
        setFailedToLaunch(true,”Failed”);
        }
}

void Dock::paintInterface(QPainter *p,
        const QStyleOptionGraphicsItem *option, const QRect &contentsRect)
{
    p->setRenderHint(QPainter::SmoothPixmapTransform);
    p->setRenderHint(QPainter::Antialiasing);
```

```
    //Here we place the code which gives a basic frame for our Dock and the
dock may look like following
}
```

void Dock::addWidget(WidgetAlias *alias,QPainter *p, const QRect &contentsRect)

{

```
        //Basic things like adding alias.svg to this dock

          Alias->svg.paint(p, (int)contentsRect.left(),(int)contentsRect.top());
          p->drawPixmap(7, 0,
        m_icon.pixmap((int)contentsRect.width(),(int)contentsRect.width()-14));
          p->save;
```

}

void Dock**::**showDock**(**QPainter *p, const QRect &contentsRect**)**

{

```
        p->restore();

        //Show the dock here
```

}

void Dock::dataUpdated(DataObject *dOb)

{

/*basic code as to what data has changed and what needs to be done e.g the mouse pointer is on the top of the screen and hence the plasmoid should become visible,or a widget is opened , this will trigger the addWidget function
*/

}

Setting up the DataEngine

DockDataEngine.desktop

//A similar .desktop file which is used in the plasmoid above.

DockDataEngine.h

```
class DockDataEgine : public Plasma::DataEngine
{
    // required since Plasma::DataEngine inherits QObject
    Q_OBJECT

    public:
        // every engine needs a constructor with these arguments
        DockDataEngine(QObject* parent, const QVariantList& args);

    protected:
        // this virtual function is called when a new source is requested
        bool sourceRequestEvent(const QString& name);

        // this virtual function is called when an automatic update
        // is triggered for an existing source (ie: when a valid update
        // interval is set when requesting a source)
        bool updateSourceEvent(const QString& source);
};
```

DockDataEngine.cpp

//This file will interact with the system to get the mouse pointer position and update data and also //keep a track of the widgets being opened and closed e.t.c and update the plasmoid it is being //used in.I have not included any code as it is all a bit hazy till now.

Tentative Timeline:

Week 1: Hacking the plasmoid APIs and trying out new things(mostly getting used to the APIs and plasmoid codes which are available.

Week 2-3: Start creating the GUI and testing things at the same time.Things like dynamically adding and removing SVG's etc.

Week 4-5: Start creating the data engine and getting knowledge about how different data from the system are collected.The trash data can easily be collected as it is a file.

Week 6-7: Interfacing the view/source.i.e the plasmoid and the dataengines.

Week 8-9:Checking the functionality.

Week 10: making changes as required and giving final touches.

Do you have other obligations from late May to early August (school, work, etc.)?:

From mid may to mid July I am completely free and will be able to give in atleast 30 hours a week toward the development. Around the 15th of July College would be starting again.After that I may be able to squeeze in around 15-20 hours in a week.

About Me (let us know who you are!):

I am an undergraduate student , currently studying in Junior Year(III year), pursuing majors in Computer Science and Engineering from IIT-BHU. One of the premier institutes of India , which takes in students who qualify IIT-JEE.

Open Source Development Experience

1>Am still working on SPLINE(Selp Paced Learning in a Networked Environment)

2>Working on Adeptran which is a open source P2P application using ADC protocol, with a discovery and lookup server.This is being made on java and is obviously cross-platform

Academic Experience

I am pursuing B.Tech Part II| with majors in Computer Science from Institute of Technology BHU, India.

I have won a few programming contests both at the college level and even at the national level.

Work Experience

Have worked on different languages including perl, ActionScript, Java, C,CPP

Have a good control over CPP