```
#
#  BlobEdit - A totally silly application for showing
#  ========   off the GUI framework and providing an example
#             of its use.
#
#  Blobs are red squares that you place on a Blob Document
#  by clicking and move around by dragging. You can save
#  your blob arrangement in a file and load it back later
#  to impress your friends (or send them away screaming).
#
#  News flash: Got a blob you don't want? Now you can
#  get rid of it by shift-clicking it! Isn't that useful!
#

import pickle
from GUI import Application, ScrollableView, Document, Window, FileType, Cursor, rgb
from GUI.Geometry import pt_in_rect, offset_rect, rects_intersect
from GUI.StdColors import black, red


class BlobApp(Application):

    def __init__(self):
        Application.__init__(self)
        self.blob_type = FileType(name = "Blob Document", suffix = "blob",
            #mac_creator = "BLBE", mac_type = "BLOB", # These are optional
        )
        self.file_type = self.blob_type
        self.blob_cursor = Cursor("blob.tiff")

    def open_app(self):
        self.new_cmd()

    def make_document(self, fileref):
        return BlobDoc(file_type = self.blob_type)

    def make_window(self, document):
        win = Window(size = (400, 400), document = document)
        view = BlobView(model = document, extent = (1000, 1000), scrolling = 'hv',
            cursor = self.blob_cursor)
        win.place(view, left = 0, top = 0, right = 0, bottom = 0, sticky = 'nsew')
        win.show()
```

```
class BlobView(ScrollableView):

    def draw(self, canvas, update_rect):
        canvas.erase_rect(update_rect)
        canvas.fillcolor = red
        canvas.pencolor = black
        for blob in self.model.blobs:
            if blob.intersects(update_rect):
                blob.draw(canvas)

    def mouse_down(self, event):
        x, y = event.position
        blob = self.model.find_blob(x, y)
        if blob:
            if not event.shift:
                self.drag_blob(blob, x, y)
            else:
                self.model.delete_blob(blob)
        else:
                self.model.add_blob(Blob(x, y))

    def drag_blob(self, blob, x0, y0):
        for event in self.track_mouse():
            x, y = event.position
            self.model.move_blob(blob, x - x0, y - y0)
            x0 = x
            y0 = y

    def blob_changed(self, model, blob):
        self.invalidate_rect(blob.rect)


class BlobDoc(Document):

    blobs = None

    def new_contents(self):
        self.blobs = []

    def read_contents(self, file):
        self.blobs = pickle.load(file)

    def write_contents(self, file):
        pickle.dump(self.blobs, file)

    def add_blob(self, blob):
```

```
        self.blobs.append(blob)
        self.changed()
        self.notify_views('blob_changed', blob)

    def find_blob(self, x, y):
        for blob in self.blobs:
            if blob.contains(x, y):
                return blob
        return None

    def move_blob(self, blob, dx, dy):
        self.notify_views('blob_changed', blob)
        blob.move(dx, dy)
        self.changed()
        self.notify_views('blob_changed', blob)

    def delete_blob(self, blob):
        self.notify_views('blob_changed', blob)
        self.blobs.remove(blob)
        self.changed()


class Blob:

    def __init__(self, x, y):
        self.rect = (x - 20, y - 20, x + 20, y + 20)

    def contains(self, x, y):
        return pt_in_rect((x, y), self.rect)

    def intersects(self, rect):
        return rects_intersect(rect, self.rect)

    def move(self, dx, dy):
        self.rect = offset_rect(self.rect, (dx, dy))

    def draw(self, canvas):
        canvas.fill_frame_rect(self.rect)


BlobApp().run()
```